IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re U.S. Patent Application of	)
	)
SUZUKI et al.	)
	)
Application Number: To be Assigned	)
	)
Filed: Concurrently Herewith	)
	)
For: DATABASE SYSTEM, SERVER, QUERY POSING METHOD, AND DATA UPDATING METHOD	)
	)
ATTORNEY DOCKET NO. HITA.0477	)

Honorable Assistant Commissioner
for Patents
Washington, D.C. 20231

REQUEST FOR PRIORITY
UNDER 35 U.S.C. § 119
AND THE INTERNATIONAL CONVENTION

Sir:

In the matter of the above-captioned application for a United States patent, notice is hereby given that the Applicant claims the priority date of March 17, 2003, the filing date of the corresponding Japanese patent application 2003-071908.

A certified copy of Japanese patent application 2003-071908 is being submitted herewith. Acknowledgment of receipt of the certified copy is respectfully requested in due course.

Respectfully submitted,

Stanley P. Fisher
Registration Number 24,344

Juan Carlos A. Marquez
Registration Number 34,072

REED SMITH LLP
3110 Fairview Park Drive
Suite 1400
Falls Church, Virginia 22042
(703) 641-4200
January 6, 2004

日 本 国 特 許 庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日
Date of Application: ２００３年 ３月１７日

出 願 番 号
Application Number: 特願２００３－０７１９０８

[ST. 10/C]: [ＪＰ２００３－０７１９０８]

出 願 人
Applicant(s): 株式会社日立製作所

２００３年 ９月２６日

特許庁長官
Commissioner,
Japan Patent Office 今井康夫

出証番号 出証特２００３－３０７９４０１

【書類名】　特許願
【整理番号】　GM0302009
【提出日】　平成15年 3月17日
【あて先】　特許庁長官殿
【国際特許分類】　G06F 9/46
G06F 15/16
【発明者】
【住所又は居所】　東京都国分寺市東恋ヶ窪一丁目２８０番地　株式会社日立製作所　中央研究所内
【氏名】　鈴木　芳生
【発明者】
【住所又は居所】　東京都国分寺市東恋ヶ窪一丁目２８０番地　株式会社日立製作所　中央研究所内
【氏名】　藤原　真二
【特許出願人】
【識別番号】　000005108
【氏名又は名称】　株式会社日立製作所
【代理人】
【識別番号】　100075513
【弁理士】
【氏名又は名称】　後藤　政喜
【選任した代理人】
【識別番号】　100084537
【弁理士】
【氏名又は名称】　松田　嘉夫
【選任した代理人】
【識別番号】　100114236
【弁理士】
【氏名又は名称】　藤井　正弘

【手数料の表示】

【予納台帳番号】　019839

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【包括委任状番号】　0110326

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　データベースシステム、サーバ、問い合わせ投入方法及びデータ更新方法

【特許請求の範囲】

【請求項１】

同一内容の検索が可能なデータベースを有し、問い合わせ要求に従って該データベースを検索する複数のデータベースサーバと、

前記問い合わせ依頼を受け付け、定められたルールを用いて前記データベースサーバに問い合わせを投入するフロントエンドサーバと、

前記フロントエンドサーバが使用するルールを管理する管理サーバと、

前記各サーバ及び問い合わせを要求するクライアント端末を接続するネットワークと、を備えるデータベースシステムであって、

前記管理サーバは、

前記データベースサーバの実行ログを取得するログ取得手段と、

前記取得したログを用いて計算された問い合わせに関する相性に基づいて前記ルールを生成するルール生成手段と、を有し、

前記フロントエンドサーバは、前記管理サーバで生成されたルールを用いて、前記問い合わせを投入する問合せ投入手段を有することを特徴とするデータベースシステム。

【請求項２】

同一内容の検索が可能なデータベースを検索する複数のデータベースサーバに対して、定められたルールを用いて、問い合わせ要求に従って問い合わせを投入するフロントエンドサーバが使用するルールを管理するサーバであって、

前記データベースサーバの実行ログを取得するログ取得手段と、

前記取得したログを用いて計算された問い合わせに関する相性に基づいて前記ルールを生成するルール生成手段と、

前記生成されたルールを用いて前記フロントエンドサーバに前記問い合わせを投入させるために、前記生成されたルールを前記フロントエンドサーバに送信するルール送信手段と、有することを特徴とするサーバ。

【請求項３】

前記ルール生成手段は、前記取得したログに含まれる問い合わせを統計的に処理することによって前記問い合わせ間の相性を計算し、ルールを生成することを特徴とする請求項２に記載のサーバ。

【請求項４】

前記ルール生成手段は、前記データベースサーバの実行ログに含まれる問い合わせをグループ化し、

グループ化された問い合わせの平均処理時間と、

該グループ化された問い合わせ及び他のグループ化された問い合わせが所定の時間関係で実行される場合の、該グループ化された問い合わせの平均処理時間と、を比較した結果に基づいて、

前記問い合わせ間の相性を計算し、ルールを生成することを特徴とする請求項３に記載のサーバ。

【請求項５】

前記ルール生成手段は、

前記データベース毎に前記実行ログ内へ記録される問い合わせの処理時間を、前記データベース間で比較した結果に基づいて、

前記問い合わせと前記データベースとの間の相性を計算し、ルールを生成することを特徴とする請求項２に記載のサーバ。

【請求項６】

前記ルール生成手段は、

あるユーザが実行した問い合わせの平均処理時間と、

該ユーザが実行する問い合わせ及び他のユーザが実行した問い合わせが、所定の時間関係で実行される場合の、該ユーザが投入した問い合わせの平均処理時間とを、比較した結果に基づいて、

問い合わせを発行するユーザ間の相性を計算し、ルールを生成することを特徴とする請求項２に記載のサーバ。

【請求項７】

前記ルール生成手段によって生成されたルールを変更するルール変更手段を有

することを特徴とする請求項２に記載のサーバ。

【請求項８】

ネットワークを介して外部からのアクセスを許容する外部アクセス手段を有し、

前記外部アクセス手段を介して、前記ルール変更手段によって前記ルールを変更することを特徴とする請求項７に記載のサーバ。

【請求項９】

前記ログ取得手段は、前記ルール送信手段によって送信されたルールに基づいて投入された問い合わせに関する実行ログを取得し、

前記ルール生成手段は、前記取得したログを用いて計算された問い合わせに関する相性に基づいて前記ルールを生成することによって、

自律的にルールを学習することを特徴とする請求項２に記載のサーバ。

【請求項１０】

同一内容の検索が可能なデータベースを有し、問い合わせ要求に従って該データベースを検索する複数のデータベースサーバと、

前記問い合わせ依頼を受け付け、定められたルールを用いて前記データベースサーバに問い合わせを投入するフロントエンドサーバと、

前記フロントエンドサーバが使用するルールを管理する管理サーバと、を備えるデータベースシステムで用いられる問い合わせ投入方法において、

前記管理サーバは、

前記データベースサーバの実行ログを取得し、

前記取得したログを用いて計算された問い合わせに関する相性に基づいて前記ルールを生成し、

前記フロントエンドサーバは、

前記管理サーバで生成されたルールを用いて、前記問い合わせを投入することを特徴とする問い合わせ投入方法。

【請求項１１】

前記取得したログに含まれる問い合わせを統計的に処理することによって前記問い合わせ間の相性を計算し、ルールを生成することを特徴とする請求項１０に

記載の問い合わせ投入方法。

【請求項１２】

前記データベースサーバの実行ログに含まれる問い合わせをグループ化し、

前記グループ化された問い合わせの平均処理時間と、

該グループ化された問い合わせ及び他のグループ化された問い合わせが所定の時間関係で実行される場合の、該グループ化された問い合わせの平均処理時間と、を比較した結果に基づいて、

前記問い合わせ間の相性を計算することを特徴とする請求項１１に記載の問い合わせ投入方法。

【請求項１３】

前記取得したログに含まれる問い合わせを統計的に処理することによって、前記問い合わせと前記データベースサーバとの間の相性を計算し、ルールを生成することを特徴とする請求項１０に記載の問い合わせ投入方法。

【請求項１４】

前記取得したログに含まれる問い合わせを統計的に処理することによって、前記問い合わせに関して、前記問い合わせを発行するユーザ間の相性を計算し、ルールを生成することを特徴とする請求項１０に記載の問い合わせ投入方法。

【請求項１５】

前記問い合わせを投入するデータベースサーバを決定する際に、該問いあわせと最も相性のよいデータベースサーバを選択することを特徴とする請求項１０に記載の問い合わせ投入方法。

【請求項１６】

前記問い合わせを投入するデータベースサーバを決定する際に、

該問いあわせと相性のよいデータベースサーバが無い場合は、

該問い合わせを投入可能なデータベースサーバの中から、該問い合わせと相性が悪いデータベースサーバサーバを除外した後に、

該問い合わせを投入するデータベースサーバを確率的に選択することを特徴とする請求項１０に記載の問い合わせ投入方法。

【請求項１７】

前記問い合わせを投入するデータベースサーバを決定する際に、

該問い合わせとの相性のよい問い合わせが直前に実行されているデータベースサーバを選択して、該問い合わせを投入することを特徴とする請求項１０に記載の問い合わせ投入方法。

【請求項１８】

前記送信されたルールに基づいて投入された問い合わせに関する実行ログを取得し、

前記取得したログを用いて計算された問い合わせに関する相性に基づいて前記ルールを生成することによって、自律的にルールを学習することを特徴とする請求項１０に記載の問い合わせ投入方法。

【請求項１９】

同一内容の検索が可能なデータベースを検索する複数のデータベースサーバに対して、定められたルールを用いて、問い合わせ要求に従って問い合わせを投入するフロントエンドサーバが使用するルールを管理する管理サーバのデータを更新するデータ更新方法であって、

ネットワークを介して外部から前記管理サーバにアクセスし、

前記管理サーバが保有するルールを取得して、

前記取得したルールに基づいて分析レポートを生成し、

前記分析レポートに基づいて、前記管理サーバが保有するルールを変更することを特徴とするデータ更新方法

【請求項２０】

前記分析レポートによって、データベースサーバの稼動の障害となるルールを検出し、

該検出されたルールを変更することを特徴とする請求項１９に記載のデータ更新方法

【発明の詳細な説明】

【０００１】

【発明が属する技術分野】

本発明は、問い合わせを実行する複数のデータベースサーバと、外部から受け

付けた問い合わせをデータベースサーバへ投入するフロントエンドサーバとから構成される検索システムに関する。

【０００２】

【従来の技術】

近年、性能や信頼性向上を目的として、データベースの並列化・分散化が進められている。図２２に、従来のデータベースシステムの典型的な構成例を示す。

【０００３】

この従来のデータベースシステムは、一つのマスターデータベース１０３に対し、複数のレプリカデータベース１０４を作成し、フロントエンドサーバ１０２は、それらのレプリカデータベース１０４に問い合わせを分散して投入することによって、データベースの検索性能の向上が図られている。また、一方を運用系とし、他方を待機系として、障害時に両者を切りかえることによって信頼性の向上が図られる場合もある。

【０００４】

このように複数のサーバに対して問い合わせを分散して投入する方法として、ラウンドロビンで割り当てるサーバを決定する方法、又は、ＣＰＵ利用率等の負荷を計測し負荷の軽いサーバへ割り当てる方法が従来から用いられている。

【０００５】

例えば、各バッチジョブのリソース使用量を計算しておき、複数のジョブを実行する場合には、累計のリソースを計算し、累計リソースがサーバの許容量を超える場合には、新たなジョブを投入しないことによってリソース競合を回避するバッチジョブのスケジューリング方法が提案されている（例えば、特許文献１参照。）。

【０００６】

データベースシステムの性能をさらに向上させるためには、レプリカデータベース１０４における、データベースバッファ競合、ディスク競合等のリソース競合を回避することが重要となる。

【０００７】

以下、図２３、図２４を用いてデータベースバッファ競合（キャッシュ競合）

について説明する。ディスク２０１はサーバ２００に接続されており、ディスク２０１には３つの表（２０５、２０６、２０７）が格納されている。表１（２０５）のデータを要求する問い合わせ１（２２０）が、サーバ２００に投入されると、ディスク２０１上の表１（２０５）から必要なデータが問い合わせに送信される。

【０００８】

ディスクの入出力（ディスクＩ／Ｏ）は、メモリの入出力に比べ、多くの処理時間を要するため、ディスクの入出力のキャッシュとしてメモリが用いられる。このキャッシュ領域（データベースバッファ）はメモリ２０３上に作成され、数ＫＢのデータページ２０４に分割されて構成されている。データページ２０４はＬＲＵ（Least Recently Used）によって、頻繁に利用されるデータほどメモリ２０３上に残るように管理される。

【０００９】

表１（２０５）のデータを取得する問い合わせ１（２２０）が発行された場合、データベースバッファが検索され、メモリ２０３上のデータベースバッファに必要とする表１（２０５）のデータが記憶されていれば、ディスクＩ／Ｏなしで結果を得ることが可能である。

【００１０】

一方、図２４（Ａ）に示すように、表２（２０６）のデータを取得する問い合わせ２（２２１）が、サーバ２００に投入されると、メモリ３００に必要なデータが記憶されていないため、ディスクＩ／Ｏが行われる。そして、図２４（Ｂ）に示すようにメモリ３０１の一部は、問い合わせの内容である表２（２０６）のデータで上書きされる。

【００１１】

例えば、それぞれの内容が異なり、かつ、大きな結果を要求する二つの問い合わせが連続して実行される場合、お互いがメモリ３００上のデータベースバッファを上書きしあい、問い合わせの実行の度にディスクＩ／Ｏが生じる。逆に、内容が共有できる問い合わせの場合は、キャッシュ（データベースバッファ）がに記憶されているデータを使用できる可能性が高く、少ないディスクＩ／Ｏで結果

を得ることができる。このように、問い合わせには相性があり、問い合わせの投入順序によって、データベースシステムの性能が変化する。

【００１２】

データベースバッファ競合（キャッシュ競合）を回避する方法としては、同じデータを要求する複数の問い合わせ間でデータベースバッファを共有させる方法がある。例えば、２つの異なる問い合わせ（クエリ）が、同一のデータを要求し、かつそのデータがデータベースバッファ領域より大きい場合に、最初のクエリがデータを途中まで読んだところで、２番目の問い合わせ（クエリ２）クエリが呼ばれると、データの前半部分はすでにデータベースバッファ上から消去されている可能性がある。すなわち、クエリが必要とするデータが大きい場合には、１番目の問い合わせ（クエリ１）が現在読んでいるデータの後半部分によって、該データの前半部分が上書きされている可能性がある。

【００１３】

このような場合、該データの前半部分は、クエリ２によってディスクから再度読み込みされ、バッファが上書きされる。しかし、上書きした部分に記憶されていたデータ（クエリ１が読みこんだデータの後半部分）は、この後クエリ２によっても必要とされるデータなので、無駄なディスクＩ／Ｏが生じる。そこで、２番目の読みこみでは、最初から読むのではなく、最初の問い合わせ（クエリ１）とバッファを共有しながら、実行中のクエリ１と同じ場所から読む方法（メリーゴーランドスキャン）がMicrosoft社のデータベース製品であるSQL Serverに導入されている（例えば、非特許文献１参照）。

【００１４】

他のバッファを有効利用する方法としては、データの重要度に応じて、キャッシュを分類する方法もある。Oracle社のデータベース製品であるOracleでは、バッファをキープ・デフォルト・リサイクルの３つの領域に分類し、領域毎にサイズや、管理方法の指定を可能としている。例えば、バッファに常駐してほしいデータは、キープへ、上書きされてもよいものはリサイクルへ、その他はデフォルトというように、バッファを分類する。但し、設定にはデータの性質についての理解が必要であり、データのサイズやシステム構成が変化する度に設定の変更が

必要となる。

【００１５】

分散化・並列化された計算機環境では、性能に加えて運用コストの増大が大きな問題となる。管理対象の計算機が増加すれば、運用コストは増大する。

【００１６】

ＩＴシステムを用いたビジネスでは、システムダウンが莫大な損失に結びつくため、システムの安定運用が必須となる。例えば、システムへのアクセス集中によるシステムダウンを回避するために、アクセス数に応じてサーバを追加することが行われる。

【００１７】

このように頻繁に構成や設定が変わる環境では、変化に追従して管理を行うことは容易ではない。環境が変化する度に、人手で計算機毎にチューニングや環境設定を実施した場合は、運用管理コストが大きくなってしまう。この運用管理コストを低下させる方法としては、自動チューニングがある。Microsoft社のSQL Serverなど、データベース単体の設定パラメータについては、自動チューニングができる製品がある。ただし、並列分散環境への対応は十分でない。

【００１８】

運用コストを低下させる他の方法として、運用管理（全部又は一部）を、ＩＴシステムの管理を請け負うＭＳＰ（Management Service Provider）を利用して、外部に委託する方法がある。例えば、ＭＳＰによって提供される監視サービスにおいては、サーバのＣＰＵ利用率などの性能情報を監視し、予め定めた閾値を超えたらアラームを発するといったサービスが実施される。

【００１９】

【特許文献１】

特開平９－３１１７９５号公報

【非特許文献１】

”データベース アーキテクチャ”、[online]、インターネット＜ＵＲＬ：http://www.microsoft.com/japan/msdn/sqlserver/sql2000/thestorageengine.asp＞

【００２０】

【発明が解決しようとする課題】

データベースシステムに投入される問い合わせには、他の問い合わせやサーバとの相性がある。相性の悪い組み合わせでは、リソース競合が生じ性能が低下する。例えば、データベースバッファの競合が発生すると、メモリに比べて速度が遅いディスクＩ／Ｏが発生し大きな性能上の問題が生じる。

【００２１】

データベース単体でみた場合には、自動チューニング技術が開発されているが、並列・分散環境でシステム全体を最適化する技術はない。

【００２２】

問い合わせを複数のサーバに分散させる方法として、例えば、ランドロビンでサーバを選択する、又は、負荷の軽いサーバを選択する方法がある。このような方法では、問い合わせ間の相性やサーバとの相性を考慮していないので、実行時にリソース競合が生じる可能性がある。そこで、並列・分散環境において、問い合わせ間や問い合わせとサーバとの間の相性を考慮して、適切なスケジューリングを実施することによって、リソース競合を回避しシステム全体のスループットを向上させることが必要となる。

【００２３】

また、サーバの並列化・分散化が進み、管理対象のサーバが多くなると、個別に細かいチューニングを行うのは困難であり高コストとなる。また、単体のサーバだけでなく、システム全体として管理することは容易ではない。そこで、並列・分散環境において、スケジューリング方法を自律的に学習し、システム全体を低コストで管理することが必要となる。

【００２４】

【課題を解決するための手段】

本発明は、同一内容の検索が可能なデータベースを有し、問い合わせ要求に従って該データベースを検索する複数のデータベースサーバと、前記問い合わせ依頼を受け付け、定められたルールを用いて前記データベースサーバに問い合わせを投入するフロントエンドサーバと、前記フロントエンドサーバが使用するルー

"ルを管理する管理サーバと、前記各サーバ及び問い合わせを要求するクライアント端末を接続するネットワークと、を備え、前記管理サーバは、前記データベースサーバの実行ログを取得するログ取得手段と、前記取得したログを用いて計算された問い合わせに関する相性に基づいて前記ルールを生成するルール生成手段と、を有し、前記フロントエンドサーバは、前記管理サーバで生成されたルールを用いて、前記問い合わせを投入する問合せ投入手段を有する。

【００２５】

【発明の作用及び効果】

本発明では、同一内容の検索が可能なデータベースを有し、問い合わせ要求に従って該データベースを検索する複数のデータベースサーバと、前記問い合わせ依頼を受け付け、定められたルールを用いて前記データベースサーバに問い合わせを投入するフロントエンドサーバと、前記フロントエンドサーバが使用するルールを管理する管理サーバと、前記各サーバ及び問い合わせを要求するクライアント端末を接続するネットワークと、を備えるデータベースシステムで用いられる問い合わせ投入方法において、前記管理サーバは、前記データベースサーバの実行ログを取得し、前記取得したログを用いて計算された問い合わせに関する相性に基づいて前記ルールを生成し、前記フロントエンドサーバは、前記管理サーバで生成されたルールを用いて、前記問い合わせを投入する。すなわち、フロントエンドサーバにキュー及びスケジューラを設け、問い合わせの間、又は、問い合わせとデータベースサーバとの間の相性を判断し、スケジューラにおいて相性のよい組み合わせで問い合わせを投入する。そしてこのスケジューリングは、ルールに基づいて行い、リソース競合を回避するようにスケジューリングがされるので、データベースシステム全体の性能向上を図ることができる。

【００２６】

また、管理サーバはデータベースサーバの実行ログを取得し、取得した実行ログを統計的に分析することによって、問い合わせ間、又は、問い合わせとデータベースサーバとの間の相性を計算し、計算した相性に基づいてルールを生成し、フロントエンドサーバに送信する。このルールは実行ログに基づいて生成されるので、環境や問い合わせの特徴が変化した場合でも自律的にルールを学習するこ

とができる。

【００２７】

【発明の実施の形態】

図１は、本発明の第１の実施の形態のデータベースシステムの構成図である。

【００２８】

クライアント１００は、ネットワーク１０１（イントラネットやインターネット（登録商標、以下同じ））を介して、データベースに対して問い合わせを発行する。

【００２９】

データベースは、データベースサーバ（フロントエンド）１０２及びバックエンドの複数のデータベースサーバ１０３、１０４によって構成される。フロントエンド１０２は、問い合わせを受け付け、バックエンドのデータベースサーバ０３、１０４に問い合わせを投入（ディスパッチ）する。

【００３０】

バックエンドのデータベースサーバは、１台のマスタデータベースサーバ（マスタ）１０３と、複数のレプリカデータベースサーバ（レプリカ）１０４によって構成される。マスタ１０３を記憶装置１０６有し、レプリカ１０４は記憶装置１０７を有する。レプリカ１０４に接続される記憶装置１０７は、マスタ１０３に接続される記憶装置１０６と同じ内容が記録されている。通常、マスタ１０３では記憶内容の更新処理及び検索処理が行われるが、レプリカ１０４では検索処理のみが行われる。そして、定期的にレプリケーションを行うことによって、マスタ１０３の記憶装置１０６の変更内容をレプリカ１０４の記憶装置１０７に反映させる。

【００３１】

レプリカ１０４は、問い合わせ実行部１１１及びログ記録・送信部１１０によって構成される。問い合わせ実行部１１１は、フロントエンド１０２が送信した問い合わせを実行する。ログ記録・送信部１１０は、実行のログ（問い合わせの種類、処理時間等）を記録し、管理サーバ１０５へ送信する。

【００３２】

管理サーバ１０５は、ログ取得部１２０、相性計算部１２１、ルール生成部１２２及びルール管理部１２３によって構成される。また、管理サーバ１０５は、ルールデータベース１４０を有する。

【００３３】

ログ取得部１２０は、レプリカ１０４が送信した実行ログを受信し、レプリカ１０４の実行ログを取得する。

【００３４】

相性計算部１２１は、ログ取得部１２０が取得した実行ログを解析し、問い合わせに関する相性として、問い合わせ間の相性や、問い合わせとレプリカ１０４との相性を計算する。相性は、実行ログに記憶される問い合わせの属性間、問い合わせの属性とレプリカとの間、問い合わせを発行したユーザ間で計算することができる。

【００３５】

ルール生成部１２２では、計算された相性に基づいて、問い合わせをスケジューリングするためのルールを生成する。例えば、問い合わせＡと問い合わせＢとは相性がよいので、同じレプリカ１０４へ投入すべきというルールが生成される。ここで、生成されたルールは、フロントエンド１０２へ送信される。

【００３６】

ルール管理部１２３は、ルール生成部１２２によって生成されたルール及びフロントエンド１０２で記録された該ルールの適用回数をルールデータベース１４０へ格納する。また、ルールデータベース１４０には、各レプリカ１０４のハードウェアやソフトウェアの性能情報（例えば、ＣＰＵ利用率、メモリ使用率等のリソース使用率）を保存することもできる。管理者は、ルール及び該ルールの適用回数に関する情報によって実際にどのルールが使用されているかを管理することが可能になる。

【００３７】

フロントエンド１０２は、キュー１３１及びスケジューラ１３０を有し、クライアント１００から受信した問い合わせをキュー１３１で受け付け、スケジューラ１３０がスケジューリングして、問い合わせ１０４を分配して、投入する。

【００３８】

スケジューラ１３０は、管理サーバ１０５が生成したルールに基づいてスケジューリングをし、問い合わせを投入するレプリカ１０４を決定する。レプリカ１０４への問い合わせの投入の際、ルール毎にそのルールが適用された回数を記録しておく。ルールの適用回数に関する情報は、管理サーバ１０５へ送信される。

【００３９】

レプリカ１０４が問い合わせを実行する。そして、レプリカ１０４において実行された問い合わせに基づいて、管理サーバ１０５が相性を計算し、ルールを生成する。そして、管理サーバにおいて生成されたルールに基づいて、フロントエンド１０２がレプリカ１０４に問い合わせを投入する。そして、レプリカ１０４が問い合わせを実行する。

【００４０】

このサイクルが繰り返されることによって、システム変更（例えば、サーバの追加、データサイズの増加）や、入力の変化（例えば、入力負荷の増加、問い合わせ内容の変化）に適応したスケジューリングルールの自律的な生成が可能になる。フロントエンド１０２において、生成されたスケジューリングルールを用いて、リソースの競合を回避する組み合わせで問い合わせを投入することによって、データベースシステム全体の性能向上を低コストで実現することが可能となる。

【００４１】

次に、レプリカ１０４、管理サーバ１０５及びフロントエンド１０２の処理のについて説明する。

【００４２】

図２は、本発明の第１の実施の形態のデータベースサーバ（レプリカ１０４）の処理を示すフローチャートである。

【００４３】

まず、クライアント１００からの問い合わせを受け付ける（ステップ４００）。そして、受け付けた問い合わせを実行する（ステップ４０１）。続いて、問い合わせの実行ログ４１０を記録し（ステップ４０２）、実行ログ４１０を管理サ

ーバ１０５へ送信する。そして、次の問い合わせを受け付ける（ステップ４００）。

【００４４】

実行ログ４１０は、問い合わせ単位でリアルタイムに送信してもよい。例えば、問い合わせの実行毎にイベントを送信し、実行ログの取得先（管理サーバ１０５）では、イベントを取得することによって、実行ログを記録する（例えば、Microsoft社のSQL Server2000）。また、複数の問い合わせの実行ログ４１０をまとめて送信してもよい。例えば、一定期間、レプリカ１０４で実行ログ４１０を記録・保存しておき、適当なタイミングで管理サーバ１０５にバッチ転送する。

【００４５】

実行ログ４１０は、図３に示す形式で送信される。すなわち、問い合わせ毎に、問い合わせ内容（ＳＱＬ文やストアドプロシージャ）７００、処理時間７０１、開始時刻７０２、終了時刻７０３、ユーザ名７０４等の問い合わせの属性を記録する。

【００４６】

また、実行ログ４１０に加えて、ハードウェアやソフトウェアのリソース使用率など、他の性能情報を管理サーバ１０５へ送ることもできる。例えば、レプリカ１０４のＣＰＵ利用率、メモリ使用率などが考えられる。これらの性能情報は、相性計算に必須ではないが、時系列データとして管理しておくことによって、レプリカ１０４の性能の監視・分析に役立つ。

【００４７】

図４は、本発明の第１の実施の形態の管理サーバ１０５の処理を示すフローチャートである。

【００４８】

レプリカ１０４から実行ログ４１０を取得する（ステップ５００）。

【００４９】

そして、相性計算の前処理として問い合わせのグループ化を行う（ステップ５０１）。相性計算は、個々の問い合わせ毎ではなく、複数の問い合わせをパターン毎に分類してグループ化したグループ毎に行う必要がある。個々の問い合わせ

毎に行うと、問い合わせの種類だけルールが生成され、該ルールは全く同一の問い合わせに対してのみ適用されるため、ルールの利用性が低くなってしまう。また、個別に計算すると、相性の組み合わせ数が多くなり、処理負担が多くなる。

【００５０】

そして、グループ化の後、グループ単位に相性を計算し（ステップ５０２）、相性計算の結果をルール化する（ステップ５０３）。最後に、生成したルール５１０をフロントエンド１０２へ送信する（ステップ５０４）。

【００５１】

フロントエンド１０２では、受信したルールに基づいてスケジューリングを実行して、適当なタイミング（予め定めた間隔、又は、管理サーバ１０５より要求があったとき）に、ルールと適用回数を管理サーバ１０５へフィードバックする。

【００５２】

管理サーバ１０５では、ルール及びルールの適用回数を受信し（ステップ５０５）、受信した情報をルールデータベース１４０へ保存する（ステップ５０６）。

【００５３】

図５は、本発明の第１の実施の形態の問い合わせのグループ化（ステップ５０１）を説明する図である。

【００５４】

以下、“select a from A where b > 3”という単純な検索ＳＱＬを用いてグループ化について説明する。ここでは、from句の引数として記述されるＡが、テーブル名である。テーブルは複数の行より構成され、行は複数の列より構成される。where句の引数では、検索条件として列名を指定する。この場合は、テーブルＡの列ｂが３より大きいという条件を指定している。どの列を結果として返すかは、select句の引数で指定する（この例では、列ａのみを取得している）。

【００５５】

図５（Ａ）に示すグループ化例１では、使用するテーブルによって、問い合わせをグループ化する。検索条件内で使用されるテーブルが同一である問い合わせ

が同一クエリにグループ化される。よって、テーブルＡのみを使用する問い合わせ１０１１及び問い合わせ１０１２は、同じグループに分類され、テーブルＢのみを使用する問い合わせ１０１３及び問い合わせ１０１４は、同じグループに分類される。

【００５６】

図５（Ｂ）に示すグループ化例２では、テーブル名及び取得列名によって問い合わせをグループ化する。検索条件内で使用されるテーブル及び検索結果として返される列名が同一である問い合わせが同一クエリにグループ化される。よって、テーブルＡの列ａを取得する問い合わせ１０２１及び問い合わせ１０２２は同じグループに分類され、テーブルＡの列ａ及びｃを取得する問い合わせ１０２３及び問い合わせ１０２４は、同じグループに分類される。このとき検索条件は異なってもよい。

【００５７】

図５（Ｃ）に示すグループ化例３では、テーブル名、取得列名及び検索条件によってグループ化する。検索条件内の定数値以外がすべて同一である問い合わせが同一クエリにグループ化される。よって、テーブルＡの列ｂを用いて条件を指定し、テーブルＡの列ａを取得する問い合わせ１０３１と問い合わせ１０３２は同じグループに分類され、テーブルＡの列ｃを用いて条件を指定し、テーブルＡの列ａを取得する問い合わせ１０３３と問い合わせ１０３４は同じグループに分類される。

【００５８】

なお、図５（Ａ）、図５（Ｂ）、図５（Ｃ）の順で詳細なグループ分けが可能になるが、グループ化のための処理量も増加する。

【００５９】

次に、相性計算（ステップ５０２）について説明する。

【００６０】

ここで、相性として、二つの問い合わせを同時に実行した場合の処理時間を、個別に実行する場合の処理時間と比べた処理時間の改善度を考える。例えば、二つの問い合わせについて検索条件内で使用されるテーブルが同じで、データベー

スバッファが共有できる場合には、これらの二つの問い合わせを同時又は連続して実行すると、処理時間を短縮することができるので、相性がよいということになる。逆に、データベースバッファが共有できない場合は、ディスク　Ｉ／Ｏによって性能が劣化するため相性が悪いということになる。

【００６１】

図６は、本発明の第１の実施の形態の問い合わせの実行状態を説明する図である。

【００６２】

図６に示す例では、Ｑｉ(900)と同じ時間にＱ２(902)が実行されている。また、Ｑ１(901)の実行時間の一部はＱｉ(900)の実行時間と重複している。

【００６３】

本実施の形態では、Ｑｉ(900)の実行と重複する問い合わせＱｊを検索する。この検索は、Ｑｊの終了時間　＞　Ｑｉ(900)の開始時間、かつ、Ｑｊの開始時間　＜　Ｑｉ(900)の終了時間、の条件を満たすＱｊを検索する。図６に示す例では、Ｑ１(901)、Ｑ２(902)がこの条件に適合する。

【００６４】

また、完全に重複する問い合わせだけを検索するのではなく、Ｑｉの実行時間の前後に余裕時間Δを加えた時間(914)と重複する問い合わせＱｊを検索してもよい。この検索は、Ｑｊの終了時間　＞　Ｑｉ(900)の開始時間－Δ(912)、かつ、Ｑｊの開始時間　＜　Ｑｉ(900)の終了時間＋Δ(913)、の条件を満たすＱｊを検索する。このように、余裕時間Δを用いて検索をすることによって、同時に実行された問い合わせに加え、Ｑｉの直前直後に実行される問い合わせも対象に含めることができる。図６に示す例では、Ｑ１(901)、Ｑ２(902)に加えて、Ｑ３(903)がこの条件に適合する。

【００６５】

図７は、本発明の第１の実施の形態において相性計算に用いられる相性マトリックスを説明する図である。

【００６６】

相性計算は、問い合わせ間の相性を示す相性マトリックスに基づいて行う。例

えば、相性マトリックス８００のＣｉｊ(803)は、問い合わせＱｉ(801)と問い合わせＱｊ(802)との相性を表し、Ｑｊ(802)と同時に実行したときの、Ｑｉ(801)の処理時間の合計値である。一方、Ａｉ(804)は、全てのＱｊ(802)との組み合わせにおけるＱｉ(801)の処理時間の合計値を表す。

【００６７】

呼出マトリックス８１０は、相性を計算するのに何個の問い合わせを使用したかを表す。例えば、Ｔｉｊ(813)は、Ｃｉｊ(803)を計算するのに何個の問い合わせを使用したのか（実行ログ中で、Ｑｉ(801)がＱｊ(802)と同時に呼び出された回数）を表す。Ｔｉ(814)は、Ａｉ(804)の計算に使用された問い合わせの数（実行ログに含まれるＱｉ(801)の実行回数）を表す。

【００６８】

図８は、本発明の第１の実施の形態の相性計算（ステップ５０２）を示すフローチャートであり、レプリカ１０４のログ毎に実行される。

【００６９】

まず、Ｑｉ(900)と同時に実行される問い合わせを検索する（ステップ１１０１）。例えば、図６において説明した方法によって検索が行われる。

【００７０】

検索条件に適合した全てのｊについて、式（１）～（３）の計算を行い、相性マトリックス８００及び呼出マトリックス８１０の値を更新する（ステップ１１０２、１１０３）。

【００７１】

Ｃｉｊ ＝ Ｃｉｊ ＋ Ｑｉの処理時間　　（１）

Ａｉ ＝ Ａｉ ＋ Ｑｉの処理時間　　（２）

Ｔｉｊ ＝ Ｔｉｊ ＋ １　　（３）

最後に、相性マトリックス８００の全要素について、式（４）によって値を更新する（ステップ１１０４、１１０５）。

【００７２】

Ｃｉｊ ＝ Ａｉ／Ｔｉ － Ｃｉｊ／Ｔｉｊ　　（４）

この式（４）の右辺は、（Ｑｉ(801)の平均処理時間）－（Ｑｊ(802)と同時に実

行した場合のＱｉ(801)の平均処理時間）を表す。つまり、Ｑｉ(801)に関して、Ｑｊ(802)と同時に実行することによる性能改善効果が計算される。この値がＱｉ(801)とＱｊ(802)との間の相性となる。

【００７３】

次に、ルールの生成（ステップ５０３）について説明する。

【００７４】

ルールの生成（ステップ５０３）では、実行ログ４１０に記録される属性やサーバに関する相性について、ルール化を行う。例えば、問い合わせ間の相性、問い合わせとサーバとの間の相性、ユーザとサーバとの間の相性などが考えられる。ここでは、例として、問い合わせ間の相性に関するルール、問い合わせとサーバの相性に関するルールの生成について説明する。

【００７５】

図９（Ａ）は、本発明の第１の実施の形態の問い合わせ間のルールの生成（ステップ５０３）を示すフローチャートである。

【００７６】

既に、レプリカ１０４毎に、相性マトリックス８００、呼出マトリックス８１０の計算が終了している。また、図９（Ａ）に示す計算は、レプリカ１０４の実行ログ４１０毎に行われる。

【００７７】

最初に、全てのＣｉｊについてルール化の対象となるか否かを判定し、ルール化の対象を絞り込む（ステップ１２０１）。具体的には、Ｃｉｊの絶対値をＡｉで除した値と、予め定めた定数との比較結果（予め定めた定数Ｐ１（０≦Ｐ１＜１）より大きいか否かの判定結果）に基づいて、Ｃｉｊをルール化の対象とするかを判定する。この条件による判定で、性能への影響が小さいＣｉｊをルール化への対象外とする。すなわち、相性の良い悪いにかかわらず、性能への影響の大きいものだけをルール化の対象とする。なお、Ｐ１＝０とした場合は、全てのＣｉｊがルール化の対象となる。

【００７８】

そして、｜Ｃｉｊ｜／Ａｉ がＰ１より大きければルール化を行い（ステップ１２

０２）、ルールリスト１へ追加する（ステップ１２０３）。

【００７９】

図１０（Ａ）に、ルールリスト１（１９１０）を示す。ルールリスト１（１９１０）では、ルールは、問い合わせの組の条件１９１１、相性値１９１２及び回数１９１３によって構成される。回数１９１３はルールが適用された回数を表し、フロントエンド１０２で設定された値であり、初期値が０を設定される。

【００８０】

ルールリスト１（１９１０）は、条件１２０１を満たす複数のルールによって構成される。すなわち、ルール化とは、Ｃｉ,ｊについて、問い合わせｉとｊの組１９１１として、Ｃｉ,ｊに対する相性値１９１２と、回数１９１３（初期値＝０）をルールリスト１へ追加することである。

【００８１】

図９（Ｂ）は、問い合わせとサーバとの間のルールの生成（ステップ５０３）を示すフローチャートである。

【００８２】

最初に、ステップ１２２０において、サーバ毎に計算されたＱｉの平均実行時間Ａｉ,ｓの平均 Ave(Ａｉ,ｓ) を計算する（ｓはサーバ名）。次に、全てのＡｉ,ｓについて、条件１２２２において、改善率 ｜Ａｉ,ｓ－Ave(Ａｉ,ｓ)｜／Ave(Ａｉ,ｓ) が予め定めた定数Ｐ２（０≦Ｐ２＜１）より大きいか否かを判定する。この条件による判定で、性能への影響が小さいＡｉ,ｓをルール化への対象外とする。すなわち、相性の良い悪いにかかわらず、性能への影響の大きいものだけをルール化の対象とする。なお、Ｐ２＝０とした場合は、全てのＡｉ,ｓがルール化の対象となる。

【００８３】

そして、｜Ａｉ,ｓ－Ave(Ａｉ,ｓ)｜／Ave(Ａｉ,ｓ) がＰ２より大きければルール化を行い（ステップ１２２３）、ルールリスト２に記録する（ステップ１２２４）。

【００８４】

図１０（Ｂ）は、ルールリスト２（１９３０）を示す。ルールリスト２（１９

３０）では、ルールは、問い合わせとサーバの組１９３１、相性値（Ave(Ａi,s)－Ａi,s）１９３２及び回数１９３３よって構成される。回数１９３４はルールが適用された回数を表し、フロントエンド１０２で設定された値であり、初期値が０に設定される。

【００８５】

ルールリスト２（１９３０）は、条件１２２２を満たす複数のルールによって構成される。すなわち、ルール化とは、Ａi,sについて、問い合わせｉとサーバｓの組１９３１として、（Ave(Ａi,s)－Ａi,s）を相性値１９３２として、回数１９３３（初期値＝０）をルールリスト２へ追加することである。

【００８６】

図１１は、本発明の第１の実施の形態において、問い合わせとサーバ（レプリカ１０４）との間の相性計算に用いられる相性マトリックスを説明する図である。

【００８７】

図７に示す相性マトリックス８００及び呼出マトリックス８１０は、レプリカ１０４毎に設けられている。相性マトリックス８００の、Ａim(804)は、レプリカｍにおいて、全てのＱj(802)との組み合わせにおけるＱi(801)の処理時間の合計値である。また、呼出マトリックス８１０の、Ｔim(814)は、レプリカｍにおいて、Ａi(804)の計算に使用された問い合わせの数（ログ中のＱi(801)の実行回数）である。

【００８８】

そして、問い合わせＱi(801)が投入されるとき、レプリカｍについての、Ａim(804)を比較して、Ａimの最小値を与えるレプリカが、Ｑiを実行するのに最も適するレプリカであるとして、レプリカと問い合わせ間の相性を判定する。

【００８９】

図１２は、本発明の第１の実施の形態の管理サーバ１０５が、ルールデータベース１４０において管理するデータファイルを説明する図である。

【００９０】

データファイル２００４は、ルール２０００、２００１、実行ログ２００３及

び性能情報２００２よって構成される。ルールは、問い合わせ間の相性を表すルールリスト１（２０００）と問い合わせとサーバの相性を表すルールリスト２（２００１）によって構成される。ルールリスト１（２０００）、実行ログ２００３、性能情報２００２は、各々レプリカ１０４の数だけ存在する。

【００９１】

実行ログ２００３は、各レプリカ１０４で実行された問い合わせに関する性能情報で、図３において前述した形式で保存される。この実行ログ２００３に基づいてルール２０００、２００１が生成される。

【００９２】

性能情報２００４は、各レプリカ１０４のハードウェアやソフトウェアに関する性能情報の時系列データであり、例えば、ＣＰＵ利用率、メモリ使用率等が記録されている。

【００９３】

これらのデータファイル２００４は、最新のデータだけ保存してもよいし、変更がある度に新たなデータを保存し、時系列データとして管理してもよい。

【００９４】

次に、フロントエンド１０２の処理について説明する。

【００９５】

図１３は、本発明の第１の実施の形態のフロントエンド１０２の処理を示すフローチャートである。

【００９６】

フロントエンド１０２では、受信したルールに基づいてスケジューリングを実行する。まず、管理サーバ１０５が送信したルール５１０（ルールリスト１、ルールリスト２）を受信する（ステップ６００）。

【００９７】

そして、ルールを編集する必要があるか否かを判定し（ステップ６０１）、必要な場合には編集を実行する（ステップ６０２）。ルールは単純なif-then型で記述されているので、人間（管理者）が理解することができる。例えば、ある問い合わせは、特定のサーバで実行してほしい等の優先すべきルールが予め分かっ

“ている場合には、ルールを追加することが可能となる。また、不要なルールを削除したり、相性値を編集することによって、特定のルールが優先的に選択されるような設定をすることもできる。

【００９８】

クライアント１００より問い合わせを受け付けると（ステップ６０３）、スケジューリングを実施し（ステップ６０４）、レプリカ１０４へ問い合わせを投入する。その際、該ルールが適用された回数を、ルール毎にルールリストの回数部１９１３、１９３３（図１０参照）へ記録する。そして、一定時間の経過や管理サーバ１０５からの要求が検出されるか否かを判定し（ステップ６０５）。ステップ６０５の条件が満たされるまでスケジューリング（ステップ６０４）が繰り返される。

【００９９】

ステップ６０５の条件が満たされたタイミングで、ルールリスト５２０（ルール及びルールの適用回数）を管理サーバへ送信する。

【０１００】

図１４は、本発明の第１の実施の形態のスケジューリング（ステップ６０４）を示すフローチャートである。

【０１０１】

ここで、各レプリカ毎の問い合わせの実行数を記録する変数として“投入数”を用意する。各レプリカ１０４毎に投入された問い合わせ数を記録しておき、投入数が所定の閾値を超えないように制御することによって、特定のレプリカ１０４に問い合わせが集中して投入されることを防ぐ意味がある。

【０１０２】

また、問い合わせを投入可能なレプリカの組として”サーバリスト”を用意する。あるレプリカ１０４に対する投入数が閾値を超えた場合は、そのレプリカをサーバリストから除外し、以降の処理で、該サーバの選択を防止する。

【０１０３】

さらに、レプリカ毎に直前に実行したクエリを記録しておく変数として“直前クエリ”を用意する。

【０１０４】

スケジューリングでは、まず最初に初期化処理を行う。各レプリカの投入数を０に初期化し、全レプリカを含むサーバリストを構成する（ステップ１３００）。そして、全レプリカの“直前クエリ”を空欄に初期化する（ステップ１３２０）。

【０１０５】

続いて、キューが空白か否かを判定する（ステップ１３０１）。キューが空白であれば、ステップ１３０１に戻り、キューが空白でなくなるまで待機する。一方、キューが空白でなければ、サーバリストに全サーバを加えた後、レプリカ毎に投入数を判定し、投入数が予め定めた閾値（Ｎ）を超える場合は、そのレプリカに問い合わせが集中して投入されているので、サーバリストから該レプリカを除外する（ステップ１３０３）。

【０１０６】

そして、キューから問い合わせを取りだし、ルールリスト１及びルールリスト２とのマッチングを行う（ステップ１３０４）。例えば、取り出された問い合わせがＱｉである場合は、｛Ｑｉ，＊｝（＊は任意）の形式のルールとマッチングを行う。

【０１０７】

ルールリスト２を用いると、＊の部分がレプリカ名（Ｓｊ）であるルールとマッチして、Ｑｉと相性が良い又は相性の悪いレプリカに関するルールが抽出される。

【０１０８】

ルールリスト１を用いると、投入可能な全てのレプリカ毎にマッチングが必要となる。最初に、“直前クエリ”変数を調査することによって、あるレプリカ（Ｒ１）の直前に投入された問い合わせ（Ｑｊ）を調べ、レプリカ（Ｒ１）のルールリスト１に｛Ｑｉ，Ｑｊ｝に関するルールがあるか否かを検索する。マッチするルールがあり、相性値が正であれば、レプリカ（Ｒ１）に関しては、ＱｉとＱｊとの相性はよいので、Ｑｉはレプリカ（Ｒ１）に投入すべきである。レプリカに対して一度も問い合わせが実行されておらず、直前の問い合わせがない場合には、

ルールリスト１を用いたマッチングは行わない。

【０１０９】

次に、相性が正であるルールがマッチしたか否かを判定する（ステップ１３０５）。そして、相性が正であるルールがある場合にはステップ１３０６へ進み、相性が正であるルールがない場合にはステップ１３０８へ進む。

【０１１０】

ステップ１３０５において相性が正であるルールがある場合には、複数のルールの中から一つのルールを選択し、そのルールが指示するレプリカへ問い合わせを投入する。選択に際しては、最も相性値の大きいルールを選択する（ステップ１３０６）。なお、最も相性値の大きいルールではなく、選択されたルールの中から確率的にルールを選択して、問い合わせを投入するレプリカ決定してもよい。

【０１１１】

そして、投入したレプリカに対して投入数を更新（１を加算）し、適用したルールの回数を更新（１を加算）する（ステップ１３０７）。そして、該レプリカの“直前クエリ”として、該ルールが指示する問い合わせを設定して（ステップ１３２６）、ステップ１３０１へ戻る。

【０１１２】

一方、ステップ１３０５において相性が正であるルールがなかった場合には、相性が負であるルールがあるか否かを判定する（条件１３０８）。そして、相性が負であるルールがある場合にはステップ１３０９へ進み、相性が負であるルールがない場合にはステップ１３１１へ進む。

【０１１３】

ステップ１３０８において相性が負であるルールがある場合には、サーバリストに含まれるレプリカから、相性が負であるルールが指示するレプリカを除外したサーバリストを作成する。そして、作成したサーバリストの中から確率的に（ランダムに）レプリカを選択し、問い合わせを投入する（ステップ１３０９）。続いて、投入したレプリカに対して投入数を更新（１を加算）し、適用したルールの回数を更新（１を加算）する（ステップ１３１０）。そして、該レプリカの

”“直前クエリ”として、該ルールが指示する問い合わせを設定して（ステップ１３２３）、条件１３０１へ戻る。

【０１１４】

一方、ステップ１３０８において相性が負であるルールがなかった場合には、サーバリストの中から確率的に（ランダムに）サーバを選択し、問い合わせを投入する。続いて、投入したレプリカに対して投入数を更新（１を加算）し、適用したルールの回数を更新（１を加算）する（ステップ１３１２）。そして、該レプリカの“直前クエリ”として、該ルールが指示する問い合わせを設定して（ステップ１３２２）、条件１３０１へ戻る。

【０１１５】

以上説明したように、第１の実施の形態では、問い合わせを実行する複数のレプリカ１０４と、レプリカ１０４に問い合わせを配分して投入するフロントエンド１０２からなるデータベースシステムにおいて、フロントエンド１０２にキュー１３１及びスケジューラ１３０を設け、問い合わせの間、又は、問い合わせとレプリカ１０４との間の相性を判断し、スケジューラ１３０において相性のよい組み合わせを抱き合わせて、レプリカ１０４に問い合わせを投入する。スケジューラ１３０は、ルールに基づいて機能し、リソース競合を回避するようにスケジューリングすることによって、リソースの競合が回避され、システム全体の性能（スループット）を向上させることができる。

【０１１６】

また、管理サーバ１０５は、レプリカ１０４の実行ログを収集し、収集した実行ログを統計解析することによって、問い合わせ間、又は、問い合わせとレプリカとの間の相性を計算する。計算した相性に基づいてルールを生成し、フロントエンド１０２に送信する。ルールは実行ログを用いて計算されるため、システム構成、問い合わせの内容や頻度が変化した場合でも自律的に学習することが可能であり、低コストな運用が可能となる。

【０１１７】

また、相性は人間にも理解可能な形式でルール化されるため、ルールの編集、追加、削除が可能となり、自律的学習に人間の意思を反映させることができる。

【０１１８】

次に、本発明の第２の実施の形態について説明する、第２の実施の形態では、第１の実施の形態における問い合わせ間及び問い合わせとサーバ間の相性に加えて、ユーザ間の相性による判定をすることもできる。その場合は、問い合わせ間の相性マトリックス８００に加え、ユーザ間の相性マトリックス２１００、２１０１を用いる。

【０１１９】

図１５に、本発明の第２の実施の形態において相性計算に用いられるユーザ間の相性マトリックスを説明する図である。

【０１２０】

相性マトリックス２１００のＣij（２１０３）は、ユーザｉ（Ｕi：２１０１）とユーザｊ（Ｕj：２１０２）との相性であり、Ｕj（２１０２）が投入した任意のクエリと同時に実行したときの、Ｕi（２１０１）が投入した任意の問い合わせの処理時間の合計値を表す。また、Ａi（２１０４）は、Ｕjにかかわらず、Ｕi（２１０１）が投入した全ての問い合わせの処理時間の合計値を表す。

【０１２１】

呼出マトリックス２１１０は、相性を計算するのに何個の問い合わせを使用したかを表す。例えば、Ｔij（２１１３）は、Ｃij（２１０３）を計算するのに何個の問い合わせを使用したのかを表す。すなわち、ログ中で、Ｕi（２１０１）が投入した問い合わせが、Ｕj（２１０２）が投入した問い合わせと同時に実行された回数である。Ｔi（２１１４）は、Ａi（２１０４）の計算に使用された問い合わせの数（Ｕi（２１０１）が問い合わせを実行した回数）を表す。

【０１２２】

問い合わせの相性と同様の方法で、ユーザ間の相性を計算し、ルールを生成し、ルールリスト３（２１３０）へルールを保存する。ルールリスト３（２１３０）では、ルールは、ユーザの組２１３１と相性値２１３１と回数２１３４によって構成される。

【０１２３】

図１６は、本発明の第２の実施の形態のユーザ間の相性を考慮する場合のスケ

ジューリング（ステップ６０４）を示すフローチャートである。なお、図１６に示すスケジューリング処理は、図１４に示すスケジューリング処理とステップ２００、２２０１、２２０２、２２０２、２２０３、２２０４において相違する。

【０１２４】

ここで、各レプリカ毎の問い合わせの実行数を記録する変数として“投入数”を、問い合わせを投入可能なレプリカの組として”サーバリスト”を、レプリカ毎に直前に実行したクエリを記録しておく変数として“直前クエリ”を用意する。さらに、該問い合わせを実行したユーザを記録する変数である“直前ユーザ”を用意する。

【０１２５】

スケジューリングでは、まず最初に初期化処理を行う。各レプリカの投入数を０に初期化し、全レプリカを含むサーバリストを構成する（ステップ１３００）。そして、全レプリカの“直前クエリ”及び”直前ユーザ”を空欄に初期化する（ステップ２２００）。

【０１２６】

続いて、キューが空白か否かを判定する（ステップ１３０１）。キューが空白であれば、ステップ１３０１に戻り、キューが空白でなくなるまで待機する。一方、キューが空白でなければ、サーバリストに全サーバを加えた後、レプリカ毎に投入数を判定し、投入数が予め定めた閾値（N）を超える場合は、そのレプリカに問い合わせが集中して投入されているので、サーバリストから該レプリカを除外する（ステップ１３０３）。

【０１２７】

そして、キューから問い合わせを取りだし、ルールリスト１及びルールリスト２とのマッチングを行う（ステップ１３０４）。例えば、キューから取り出した問い合わせを投入したユーザ（Uiとする）についてマッチングを行う。このマッチングにおいては、レプリカ毎にルールリスト３に条件部が｛Ui，“直前ユーザ”｝となるルールがあるかを調べる。すなわち、ユーザ（Ui）が投入する問い合わせと相性が良い問い合わせを投入するユーザがいるか、又は、相性が悪

い問い合わせを投入するユーザがいるかを判定する。

【０１２８】

次に、相性が正であるルールがマッチしたか否かを判定する（ステップ１３０５）。そして、相性が正であるルールがある場合にはステップ１３０６へ進み、相性が正であるルールがない場合にはステップ１３０８へ進む。

【０１２９】

ステップ１３０５において相性が正であるルールがある場合には、複数のルールの中から一つのルールを選択し、そのルールが指示するレプリカへ問い合わせを投入する。選択に際しては、最も相性値の大きいルールを選択する（ステップ１３０６）。なお、最も相性値の大きいルールではなく、選択されたルールの中から確率的にルールを選択して、問い合わせを投入するレプリカ決定してもよい。

【０１３０】

そして、投入したレプリカに対して投入数を更新（１を加算）し、適用したルールの回数を更新（１を加算）する（ステップ１３０７）。そして、該レプリカの“直前クエリ”として、該ルールが指示する問い合わせを設定し、“直前ユーザ”として、該問い合わせを投入したユーザを設定して（ステップ２２０４）、ステップ１３０１へ戻る。

【０１３１】

一方、ステップ１３０５において相性が正であるルールがなかった場合には、相性が負であるルールがあるか否かを判定する（条件１３０８）。そして、相性が負であるルールがある場合にはステップ１３０９へ進み、相性が負であるルールがない場合にはステップ１３１１へ進む。

【０１３２】

ステップ１３０８において相性が負であるルールがある場合には、サーバリストに含まれるレプリカから、相性が負であるルールが指示するレプリカを除外したサーバリストを作成する。そして、作成したサーバリストの中から確率的に（ランダムに）レプリカを選択し、問い合わせを投入する（ステップ１３０９）。続いて、投入したレプリカに対して投入数を更新（１を加算）し、適用したルー

ルの回数を更新（１を加算）する（ステップ１３１０）。そして、該レプリカの“直前クエリ”として、該ルールが指示する問い合わせを設定し、“直前ユーザ”として、該問い合わせを投入したユーザを設定して（ステップ２２０３）、ステップ１３０１へ戻る。

【０１３３】

一方、ステップ１３０８において相性が負であるルールがなかった場合には、サーバリストの中から確率的に（ランダムに）サーバを選択し、問い合わせを投入する。続いて、投入したレプリカに対して投入数を更新（１を加算）し、適用したルールの回数を更新（１を加算）する（ステップ１３１２）。そして、該レプリカの“直前クエリ”として、該ルールが指示する問い合わせを設定し、“直前ユーザ”として、該問い合わせを投入したユーザを設定して（ステップ２２０２）、ステップ１３０１へ戻る。

【０１３４】

以上説明したように、第２の実施の形態では、問い合わせを実行する複数のレプリカ１０４と、レプリカ１０４に問い合わせを配分して投入するフロントエンド１０２からなるデータベースシステムにおいて、フロントエンド１０２にキュー１３１及びスケジューラ１３０を設け、問い合わせを投入するユーザ間の相性、又は、問い合わせを投入するユーザとレプリカ１０４との相性を判断し、スケジューラ１３０において相性のよい組み合わせを抱き合わせて、レプリカ１０４に問い合わせを投入する。スケジューラ１３０は、ルールに基づいて機能し、リソース競合を回避するようにスケジューリングすることによって、リソースの競合が回避され、システム全体の性能（スループット）を向上させることができる。

【０１３５】

次に、本発明の第３の実施の形態について説明する、第３の実施の形態においては、外部サーバをネットワーク経由で管理サーバに接続し、外部サーバから管理サーバへのアクセスを可能とすることによって、外部から監視、管理を行うことができる。

【０１３６】

図１７は、本発明の第３の実施の形態のデータベースシステムの構成図である。

【０１３７】

管理サーバ１０５はネットワーク１６０１（イントラネットやインターネット）に接続される。管理サーバ１０５はネットワーク１６０１を介して、管理を請け負うＭＳＰ（Management Service Provider）の監視・分析サーバへ接続１６００される。管理サーバ１０５のルールデータベース１４０には、ルールや性能データが蓄積されており、監視・分析サーバ１６００でそれらを分析することによって、外部からのリモート保守やシステムの診断をすることができる。

【０１３８】

監視・分析サーバ１６００は、性能情報管理部１６０２及び分析レポート生成部１６０２よって構成される。性能情報管理部１６０２では、管理サーバ１０５から、ルールやルールの適用回数、レプリカ毎のリソース使用率等を、ルール・性能情報１６１０として取得する。得られたルール・性能情報１６１０は、ルール・性能情報データベース１６２０へ格納される。ルール・性能情報データベース１６２０は、時系列で情報を管理する。分析レポート生成部１６０３では、性能情報に基づいて分析レポート１６１２を生成する。

【０１３９】

また、監視・分析サーバ１６００によって、管理サーバ１０５のリモート保守１６１３をする。

【０１４０】

情報システムの経営者又は運営者は、その監視分析の対価としてサービス料、保守料１６１１をＭＳＰへ払う。情報システムの経営者又は運営者は、データベース管理（又は、その一部）を外部委託することが可能となる。

【０１４１】

図１８は、本発明の第３の実施の形態の監視・分析サーバ１６００と管理サーバ１０５との間の処理のフローチャートを示す。図中左側が監視・分析サーバ１６００の処理を、右側が管理サーバ１０５の処理である。

【０１４２】

監視・分析サーバ１６００は、管理サーバ１０５へアクセス要求を送信する（ステップ１４００）。

【０１４３】

管理サーバ１０５は、監視・分析サーバ１６００からのアクセスを受け付け、認証を行う（ステップ１４１０）。認証が成功した場合は（ステップ１４１１）、ルールデータベース１４０に記録されるルール・性能情報を送信する（ステップ１４１２）。

【０１４４】

要求（１４００）が認められた監視・分析サーバ１６００は、ルール・性能情報１６１０を取得する（ステップ１４０１）。取得したルール・性能情報１６１０は、ルール・性能情報データベース１６２０に格納される。

【０１４５】

そして、ルール・性能情報データベース１６２０に格納される性能情報について、後述する方法によって時系列解析をして、分析レポートを生成する（ステップ１４０２）。最後に、生成した分析レポートをメールで送信する（ステップ１４０３）。なお、生成した分析レポートを監視・分析サーバ１６００（又は、管理サーバ１０５）がネットワークを介してアクセス可能なｗｅｂサーバ（図示省略）に格納すると共に、分析レポートが作成されたことを管理サーバ１０５へ通知してもよい。

【０１４６】

そして、管理サーバ１０５は分析レポートを受信する（ステップ１４１３）。

【０１４７】

その後、監視・分析サーバ１６００では、ルールに問題（例えば、ある特定のルールに問題があるためのスループットの低下）があるかを判定する（ステップ１４０４）。そして、問題となるルールを編集する（ステップ１４０５）。例えば、問題となるルールを削除したり、該ルールの相性値を変更することによって、ルールの適用を制御して、スループットを向上させる。そして、編集されたルールを管理サーバ１０５に送信する（ステップ１４０６）ことによってルールの保守を行う。

【０１４８】

管理サーバ１０５は、監視・分析サーバ１６００から送信されたルールを受け付け、ルールデータベース１４０に記憶する（ステップ１４１４）。そして、ルールをフロントエンド１０２へ送信し（ステップ１４１５）、新しいルールでのスケジューリングを指示する。

【０１４９】

図１９は、本発明の第３の実施の形態の監視・分析サーバ１６００が、性能情報データベース１６２０において管理するデータファイルを説明する図である。

【０１５０】

データファイル２００５は、ルール２０００、２００１及び性能情報２００２よって構成される。ルールは、ルールリスト１（２０００）及びルールリスト２（２００１）よって構成される。ルールリスト１（２０００）及び性能情報（２００２）は、各々レプリカ１０４の数だけ存在する。

【０１５１】

また、データファイル２００５は、ルールが変更される度に新たに生成され、時系列データとして管理される。

【０１５２】

図２０は、本発明の第３の実施の形態の分析レポートの表示画面の例を説明する図である。

【０１５３】

分析レポート１５００として、スループット表示１５０１、ルール表示１５０２、性能表示１５０３が表示されている。

【０１５４】

スループット表示１５０１としては、システム全体の性能のグラフが表示される。例えば、単位時間あたりの問い合わせ処理量を表示する。スループット表示１５０１によって、システム全体の性能の時系列変化を把握することができる。また、グラフを外挿することによって、将来のスループットを予測することもできる。

【０１５５】

ルール表示１５０２では、例えば、ある時間にフロントエンドにおいてスケジューリングに使用されたルールに関する情報を表示する。ルールを解析する期間（時間）は、ユーザが直接入力してもよいし、スループット画面１５０１で指定してもよい。ルール表示１５０２では、ルールの条件部、相性値、適用回数が表示される。ルール表示１５０２でルールの条件部に表示されるのは問い合わせにつけられたＩＤであり、問い合わせの内容ではないため、ルール詳細表示１５１０を別に設け、実際のルールの内容を表示する。ルール表示１５０４によって、どのようなルールが生成され、それぞれ何回適用されているのかを把握することができる。

【０１５６】

また、適用されるルールの時系列的な変化を知ることもできる。例えば、ルールは学習され変化していくため、スループットが低下している場合は、適用されるルールに変化があるかどうかをチェックすることによって、スループット低下の原因を調査することができる。ある特定のルールに問題がある場合は、ルールを編集することによって保守を行う。例えば、該ルールを削除したり、該ルールの相性値を変更することによって、ルールの適用を制御して、スループットを向上させることができる。

【０１５７】

性能画面１５０３は、レプリカ毎にＣＰＵ利用率、ディスク利用率といった性能リソースに関する情報をグラフ表示する。性能画面によってレプリカ毎の負荷や、レプリカ間の負荷バランスを把握することができる。この性能画面によって、レプリカ毎のボトルネックが分かれば、そのボトルネックを回避するための設備投資提案をすることが可能である。例えば、メモリの使用率が高くなっていれば、メモリ増設を提案することができる。

【０１５８】

図２１は、第３の実施の形態による監視、管理サービスのビジネスモデルの概略を説明する図である。

【０１５９】

データベースシステム（例えば、図１）等のＩＴシステム１８００のシステム

の管理者又は経営者は、外部のＭＳＰ１８０１と契約を結び管理をアウトソーシングする。

【０１６０】

ＩＴシステム１８００からは、ルール・性能情報１８０２をＭＳＰ１８０１へ送信する。このルール・性能情報１８０２は、サーバのリソース使用率といった性能情報に加え、ＩＴシステム１８００において自律的に生成されたルールも含む。

【０１６１】

ＭＳＰ１８０１では、従来行われてきたリソース使用率等の性能情報の監視・分析に加え、ルールの監視・分析を実施する。ＭＳＰ１８０１では、ルール・性能情報１８０２を受けて、分析を実施して分析結果をレポート１８０３にまとめ、ＩＴシステム１８００へ提供する。また、ＭＳＰ１８０１では、リモート保守１８０５を行う。ここでは、従来のソフトウェアの設定パラメータの変更等に加え、ルールの編集を行う。ルールは、複数のデータを抽象化して（例えば、if-then形式で）記載されており、可読性が高いため人間による理解も容易であることから、ＭＳＰ１８０１の管理者によって編集をすることができる。

【０１６２】

また、ＭＳＰ１８０１による診断の上、レプリカ１０４等のハードウェアの変更提案も可能である。ルール・性能情報１８０２を時系列で管理することによって、システムの変化を把握すれば、性能上の問題が発生する前に対策を取ることができる。

【０１６３】

ＩＴシステム１８００の管理者又は経営者は、保守やサービスに対し対価１８０４をＭＳＰ１８０１に支払う。

【０１６４】

より具体的には、ＭＳＰ１８０１では、分析レポート１８０３を定期的に実行する。ユーザである、ＩＴシステム１８００の管理者又は経営者は、その対価として、サポート・保守料をＭＳＰ１８０１に支払う。

【０１６５】

以上説明したように、第３の実施の形態では、外部の監視・分析サーバ１６００によって、管理サーバ１０５の監視・管理をするので、抽象化されたルールを扱うことによって、システムの挙動の把握が容易であり、ルール変更による保守も容易であるため、高付加価値サービスの提供が可能である。

【図面の簡単な説明】

【図１】

本発明の第１の実施の形態のデータベースシステムの構成図である。

【図２】

本発明の第１の実施の形態のレプリカ１０４の処理のフローチャートである。

【図３】

本発明の第１の実施の形態のレプリカ１０４で記録されるログデータの構成図である。

【図４】

本発明の第１の実施の形態の管理サーバ１０５の処理のフローチャートである。

【図５】

本発明の第１の実施の形態の問い合わせのグループ化の説明図である。

【図６】

本発明の第１の実施の形態の問い合わせの実行状態の説明図である。

【図７】

本発明の第１の実施の形態において相性計算に用いられる相性マトリックスの説明図である。

【図８】

本発明の第１の実施の形態の相性計算（ステップ５０２）のフローチャートである。

【図９】

本発明の第１の実施の形態の問い合わせ間のルールの生成（ステップ５０３）のフローチャートである。

【図１０】

本発明の第１の実施の形態のルールリストの構成図である。

【図１１】

本発明の第１の実施の形態において相性計算に用いられる別の相性マトリックスの説明図である。

【図１２】

本発明の第１の実施の形態の管理サーバ１０５がルールデータベース１４０において管理するデータファイルの説明図である。

【図１３】

本発明の第１の実施の形態のフロントエンド１０２の処理のフローチャートである。

【図１４】

本発明の第１の実施の形態のスケジューリング（ステップ６０４）のフローチャートである。

【図１５】

本発明の第２の実施の形態において相性計算に用いられるユーザ間の相性マトリックスの説明図である。

【図１６】

本発明の第２の実施の形態のスケジューリング（ステップ６０４）のフローチャートである。

【図１７】

本発明の第３の実施の形態のデータベースシステムの構成図である。

【図１８】

本発明の第３の実施の形態の監視・分析サーバ１６００と管理サーバ１０５との間の処理のフローチャートである

【図１９】

本発明の第３の実施の形態の監視・分析サーバ１６００が、性能情報データベース１６２０において管理するデータファイルの説明図である。

【図２０】

本発明の第３の実施の形態の分析レポートの表示画面の例を説明する図である

。

【図２１】

監視、管理サービスのビジネスモデルの概略図である。

【図２２】

従来のデータベースシステムの構成図である。

【図２３】

従来のデータベースシステムのデータベースバッファの動作の説明図である。

【図２４】

従来のデータベースシステムのデータベースバッファの動作の説明図である。

【符号の説明】

１００　クライアント

１０１　ネットワーク

１０２　データベースサーバ（フロントエンド）

１０３　データベースサーバ（マスタ）

１０４　データベースサーバ（レプリカ）

１０５　管理サーバ

１０６　マスタディスク

１０７　レプリカディスク

１１０　ログ記録・送信部

１１１　問い合わせ実行部

１２０　ログ取得部

１２１　相性計算部

１２２　ルール生成部

１２３　ルール管理部

１４０　ルールデータベース

２００　サーバ

２０１　ディスク

２０３　メモリ

１６００　監視・分析サーバ

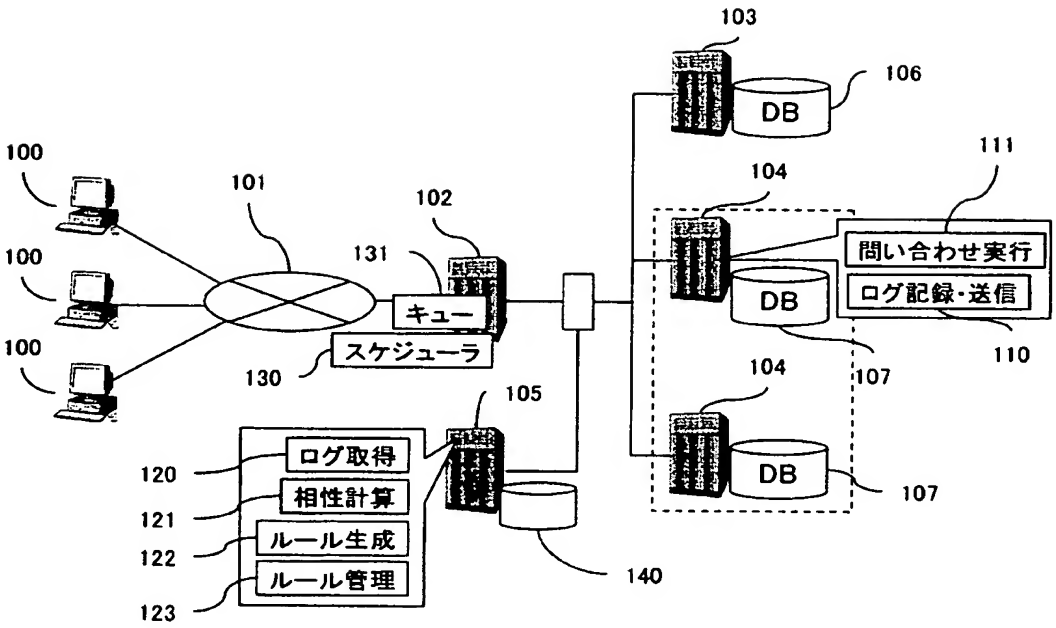
１６０１　ネットワーク

１６０２　性能情報管理部

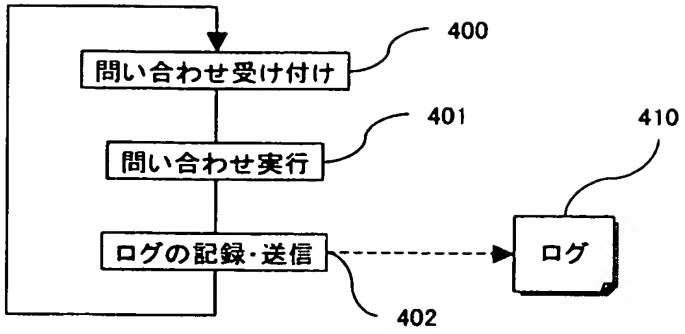
１６０３　分析レポート生成部

【書類名】　図面

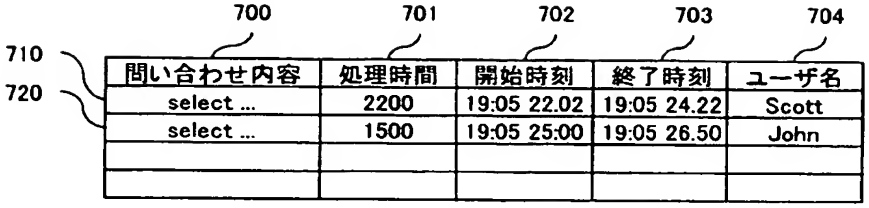
【図１】


103
106
DB
100
101
102
104
111
問い合わせ実行
131
キュー
DB
ログ記録・送信
100
スケジューラ
130
110
107
104
100
105
ログ取得
120
相性計算
121
DB
107
ルール生成
122
ルール管理
123
140

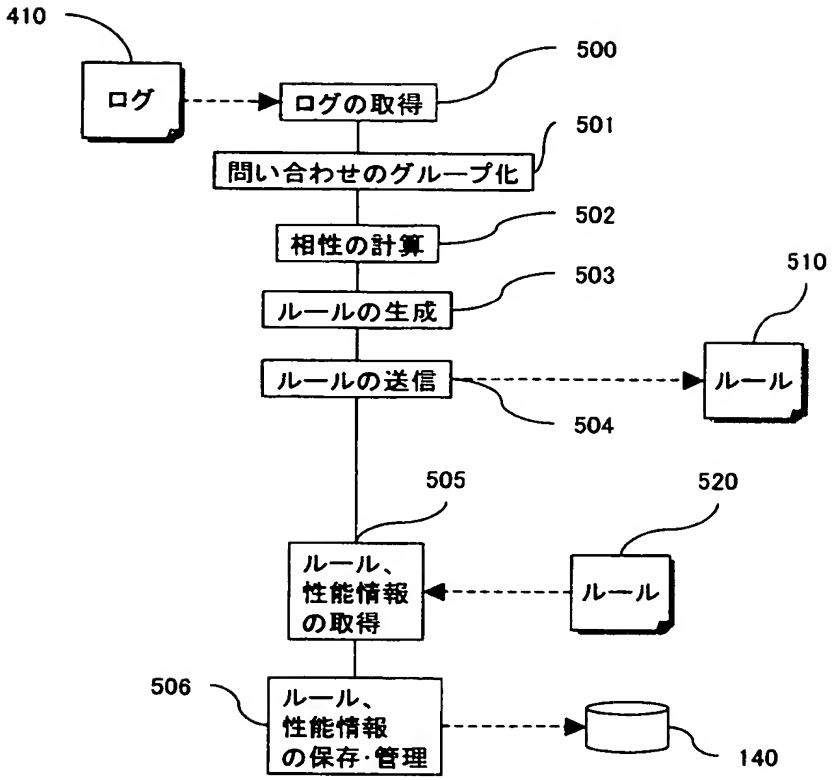
【図２】



【図３】

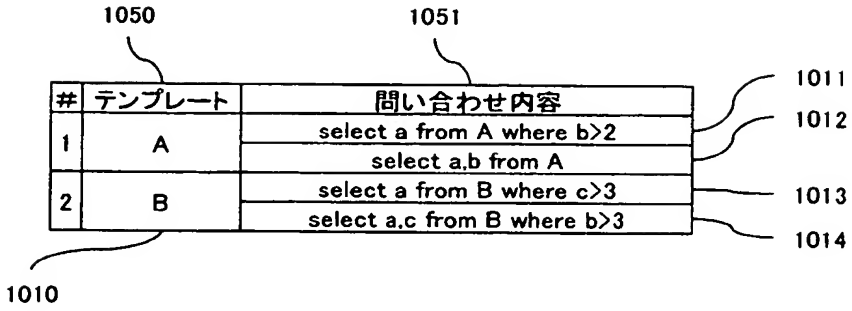

700
701
702
703
704
710
720
問い合わせ内容
処理時間
開始時刻
終了時刻
ユーザ名
select ...
2200
19:05 22.02
19:05 24.22
Scott
select ...
1500
19:05 25:00
19:05 26.50
John

【図４】

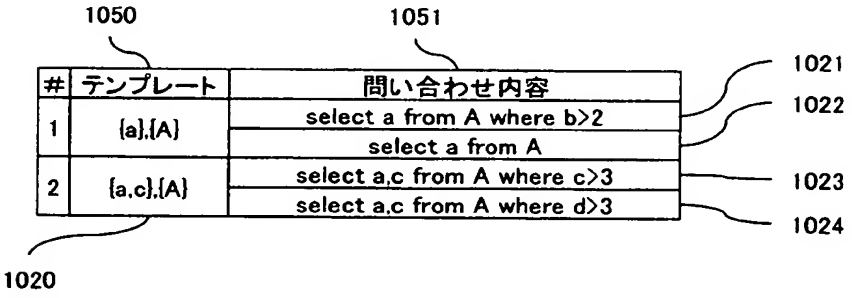

410
ログ
ログの取得
500
問い合わせのグループ化
501
相性の計算
502
ルールの生成
503
510
ルール
ルールの送信
504
505
520
ルール、
性能情報
の取得
ルール
506
ルール、
性能情報
の保存･管理
140

【図５】

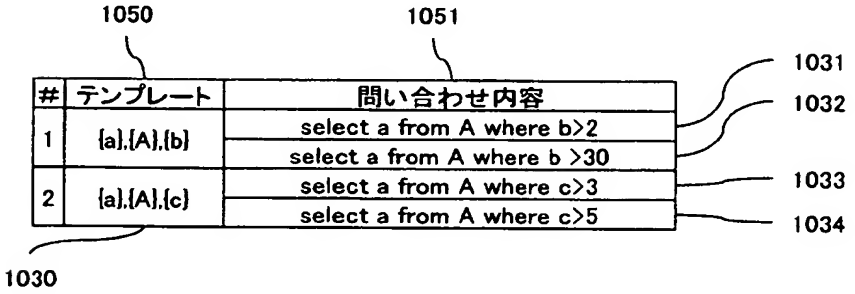
(A) グループ化例1



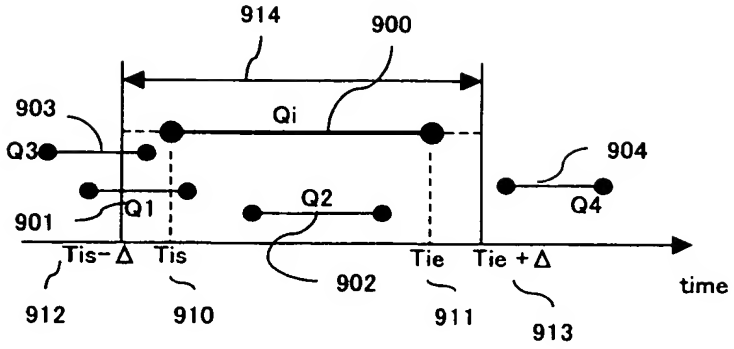
(B) グループ化例2



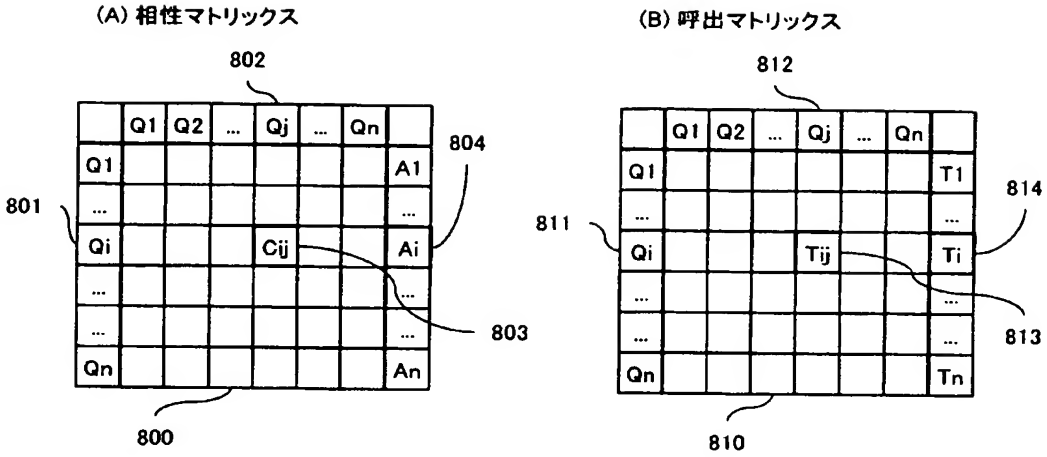
(C) グループ化例3



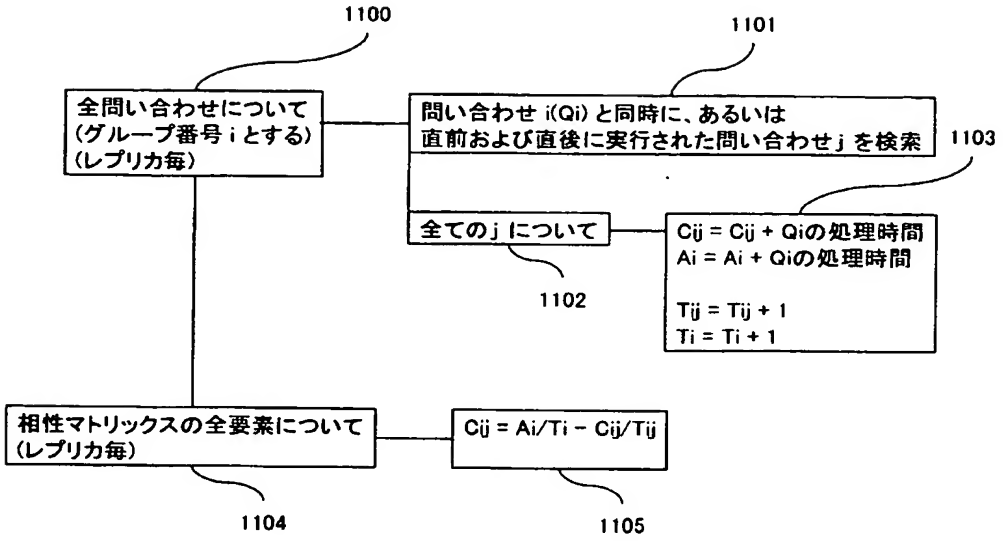
【図６】


914
900
903
Qi
Q3
904
Q1
Q2
Q4
901
Tis－Δ
Tis
Tie
Tie＋Δ
time
912
910
902
911
913

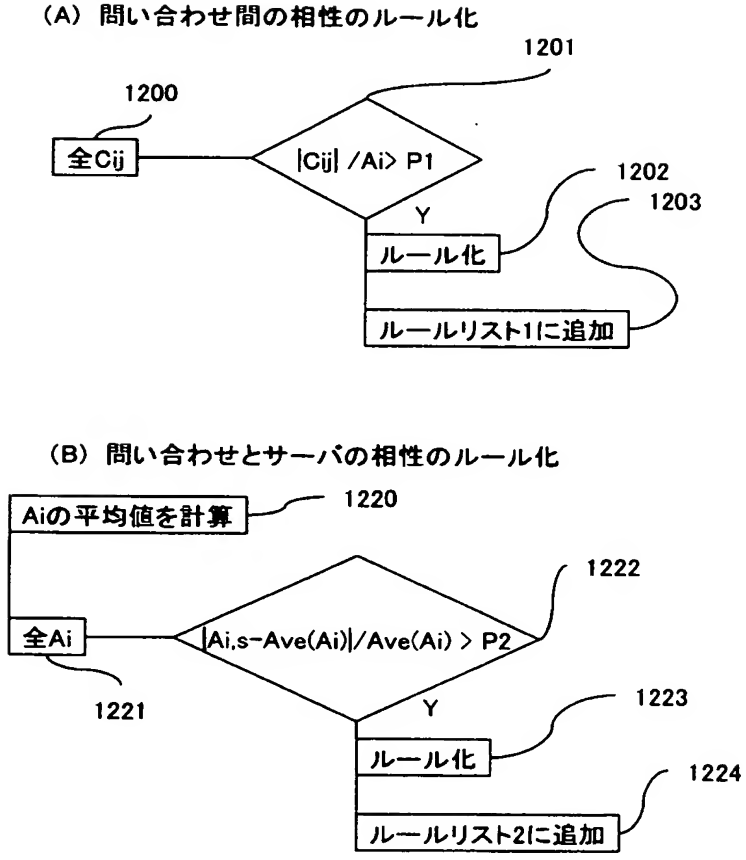
【図７】


(A) 相性マトリックス
802
Q1 Q2 ... Qj ... Qn
Q1
...
Qi
...
...
Qn
A1
...
Ai
...
...
An
804
801
Cij
803
800
(B) 呼出マトリックス
812
Q1 Q2 ... Qj ... Qn
Q1
...
Qi
...
...
Qn
T1
...
Ti
...
...
Tn
814
811
Tij
813
810

【図８】

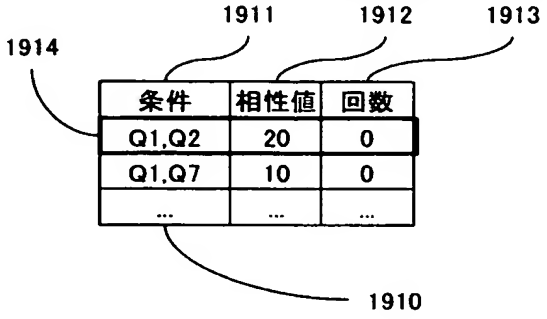

1100
全問い合わせについて
(グループ番号 i とする)
(レプリカ毎)
1101
問い合わせ i(Qi) と同時に、あるいは
直前および直後に実行された問い合わせ j を検索
1103
全ての j について
1102
Cij = Cij + Qiの処理時間
Ai = Ai + Qiの処理時間
Tij = Tij + 1
Ti = Ti + 1
相性マトリックスの全要素について
(レプリカ毎)
1104
Cij = Ai/Ti − Cij/Tij
1105

【図９】

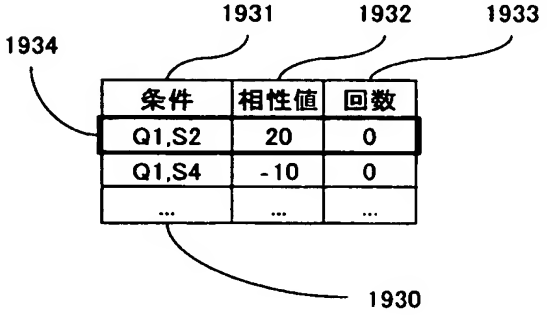

(A) 問い合わせ間の相性のルール化
1201
1200
全Cij
|Cij| /Ai> P1
1202
1203
Y
ルール化
ルールリスト1に追加
(B) 問い合わせとサーバの相性のルール化
Aiの平均値を計算
1220
1222
全Ai
|Ai,s-Ave(Ai)|/Ave(Ai) > P2
1221
Y
1223
ルール化
1224
ルールリスト2に追加

【図１０】

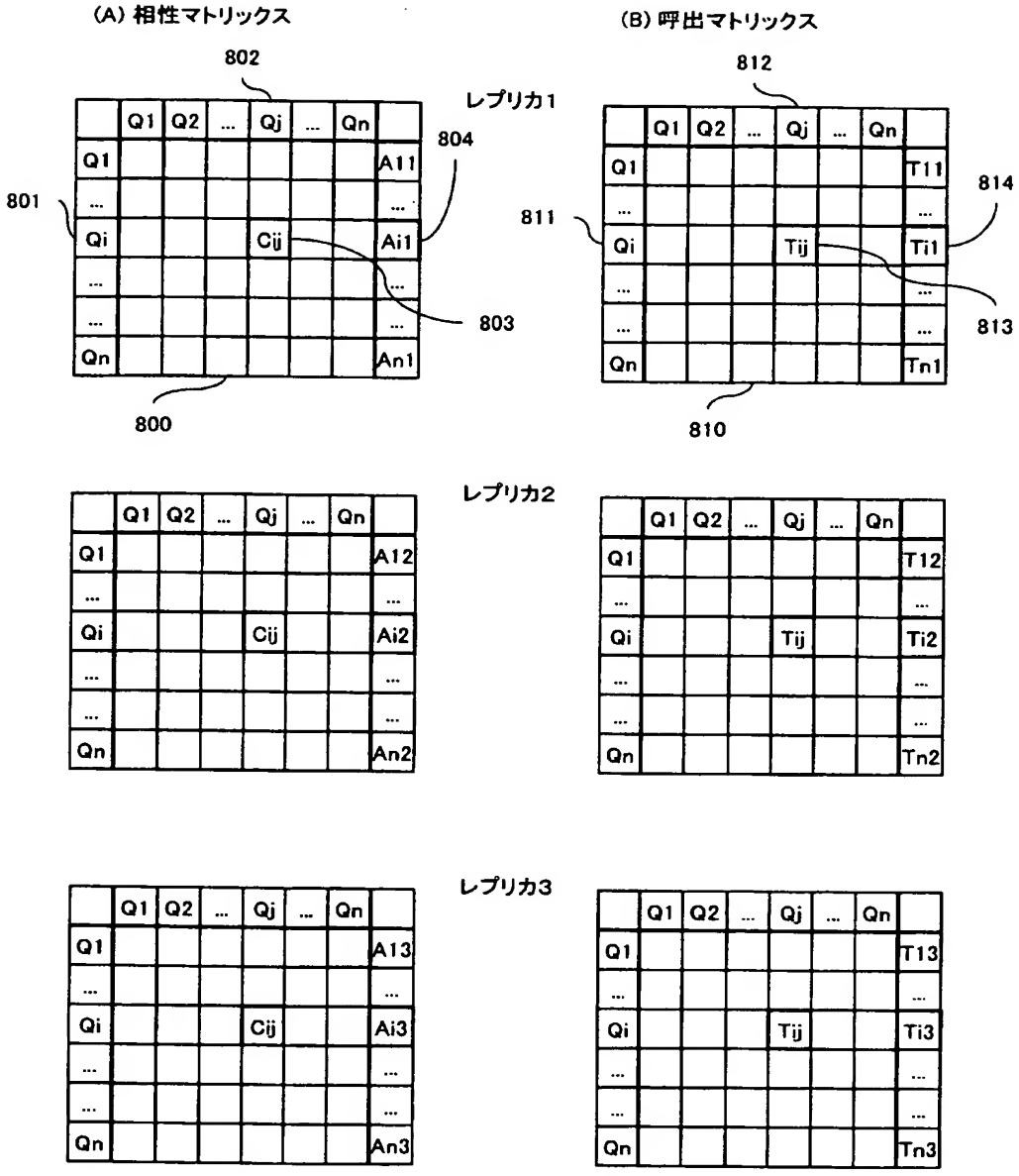
(A) ルールリスト1(問い合わせ間の相性のルール)



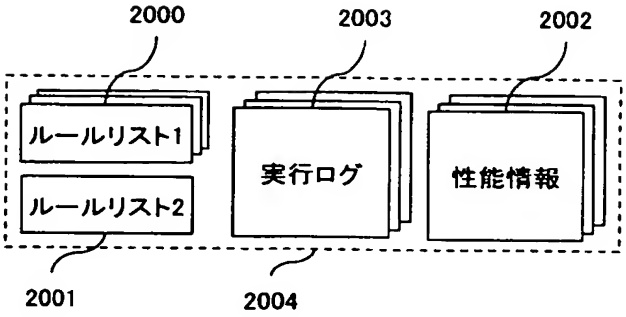
(B) ルールリスト2(問い合わせとサーバの相性のルール)



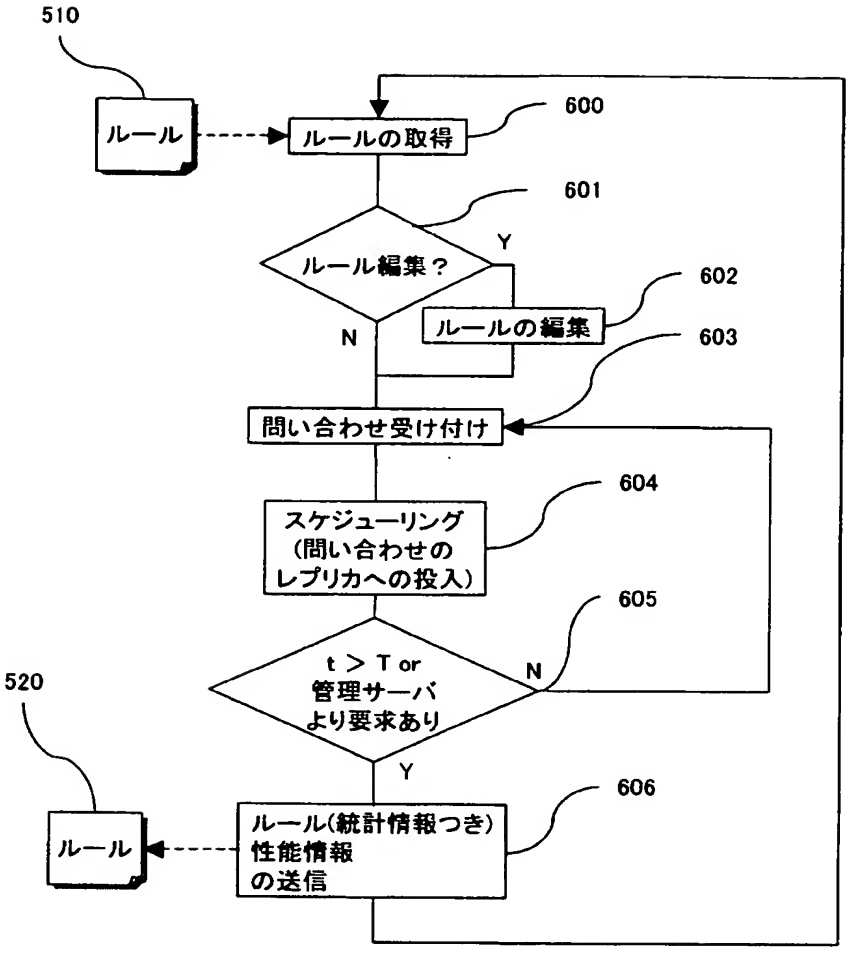
【図１１】


(A) 相性マトリックス
(B) 呼出マトリックス
802
812
レプリカ1
804
801
811
814
803
813
800
810
レプリカ2
レプリカ3
Q1
Q2
...
Qj
Qn
Qi
Cij
Tij
A11
A12
A13
Ai1
Ai2
Ai3
An1
An2
An3
T11
T12
T13
Ti1
Ti2
Ti3
Tn1
Tn2
Tn3

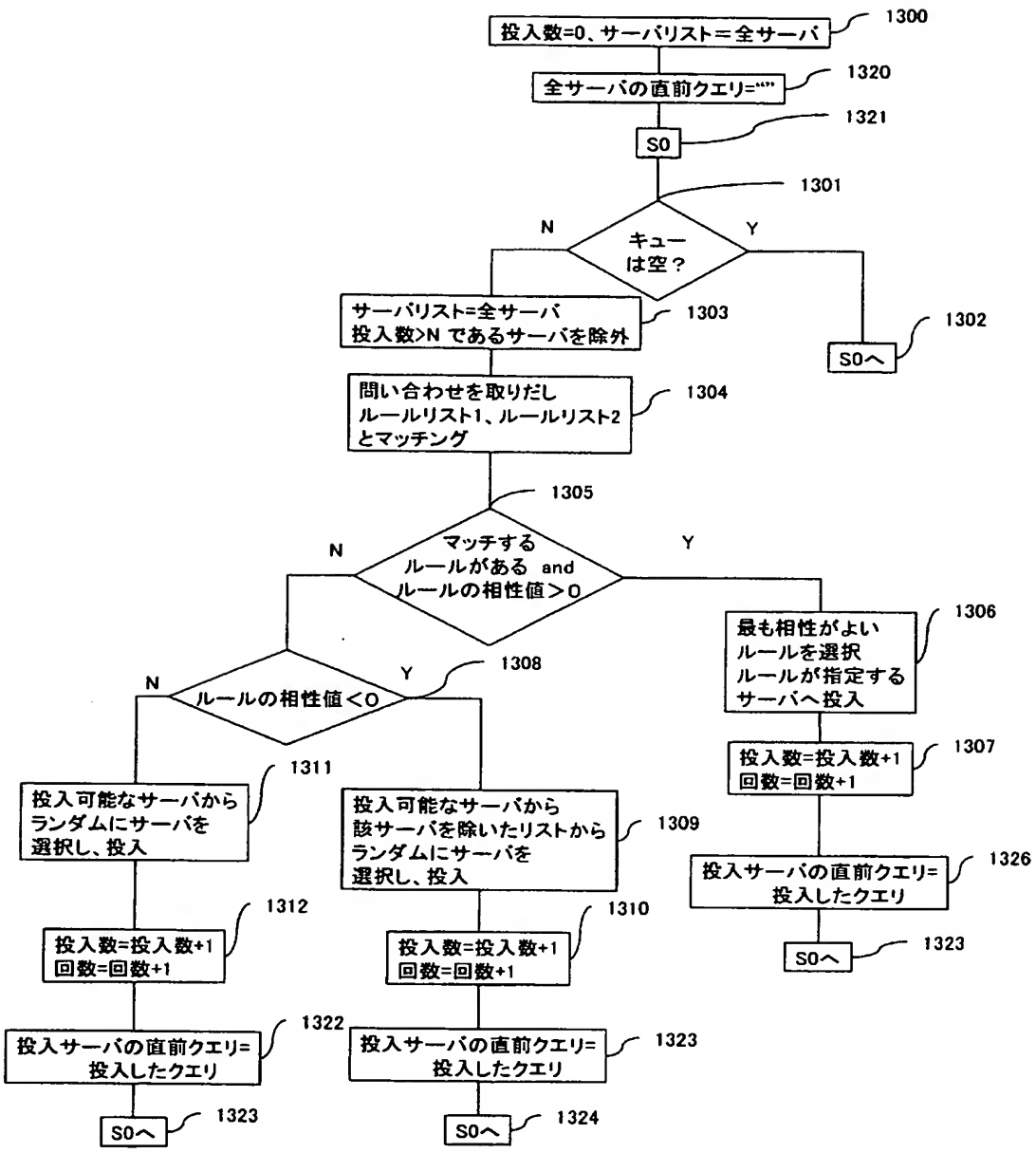
【図１２】



【図１３】

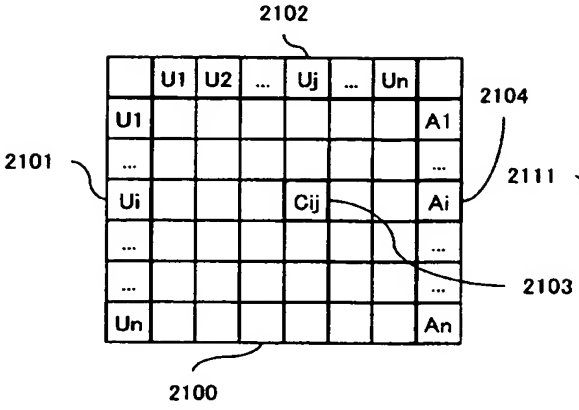

510
ルール
ルールの取得
600
601
ルール編集？
Y
N
602
ルールの編集
603
問い合わせ受け付け
604
スケジューリング
(問い合わせの
レプリカへの投入)
605
t > T or
管理サーバ
より要求あり
N
Y
520
ルール
606
ルール(統計情報つき)
性能情報
の送信

【図１４】

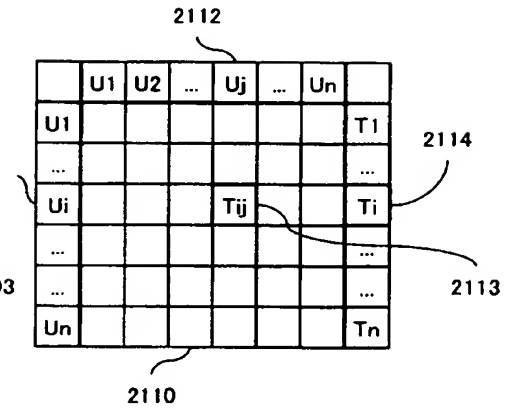

投入数=0、サーバリスト＝全サーバ
1300
全サーバの直前クエリ=""
1320
S0
1321
1301
キュー
は空？
N
Y
サーバリスト=全サーバ
投入数>N であるサーバを除外
1303
S0へ
1302
問い合わせを取りだし
ルールリスト1、ルールリスト2
とマッチング
1304
1305
マッチする
ルールがある and
ルールの相性値＞0
N
Y
最も相性がよい
ルールを選択
ルールが指定する
サーバへ投入
1306
投入数=投入数+1
回数=回数+1
1307
投入サーバの直前クエリ=
投入したクエリ
1326
S0へ
1323
ルールの相性値＜0
N
Y
1308
投入可能なサーバから
ランダムにサーバを
選択し、投入
1311
投入可能なサーバから
該サーバを除いたリストから
ランダムにサーバを
選択し、投入
1309
投入数=投入数+1
回数=回数+1
1312
投入数=投入数+1
回数=回数+1
1310
投入サーバの直前クエリ=
投入したクエリ
1322
投入サーバの直前クエリ=
投入したクエリ
1323
S0へ
1323
S0へ
1324

【図１５】

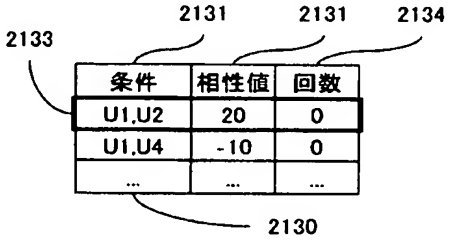
（A）相関マトリックス



（B）呼出マトリックス

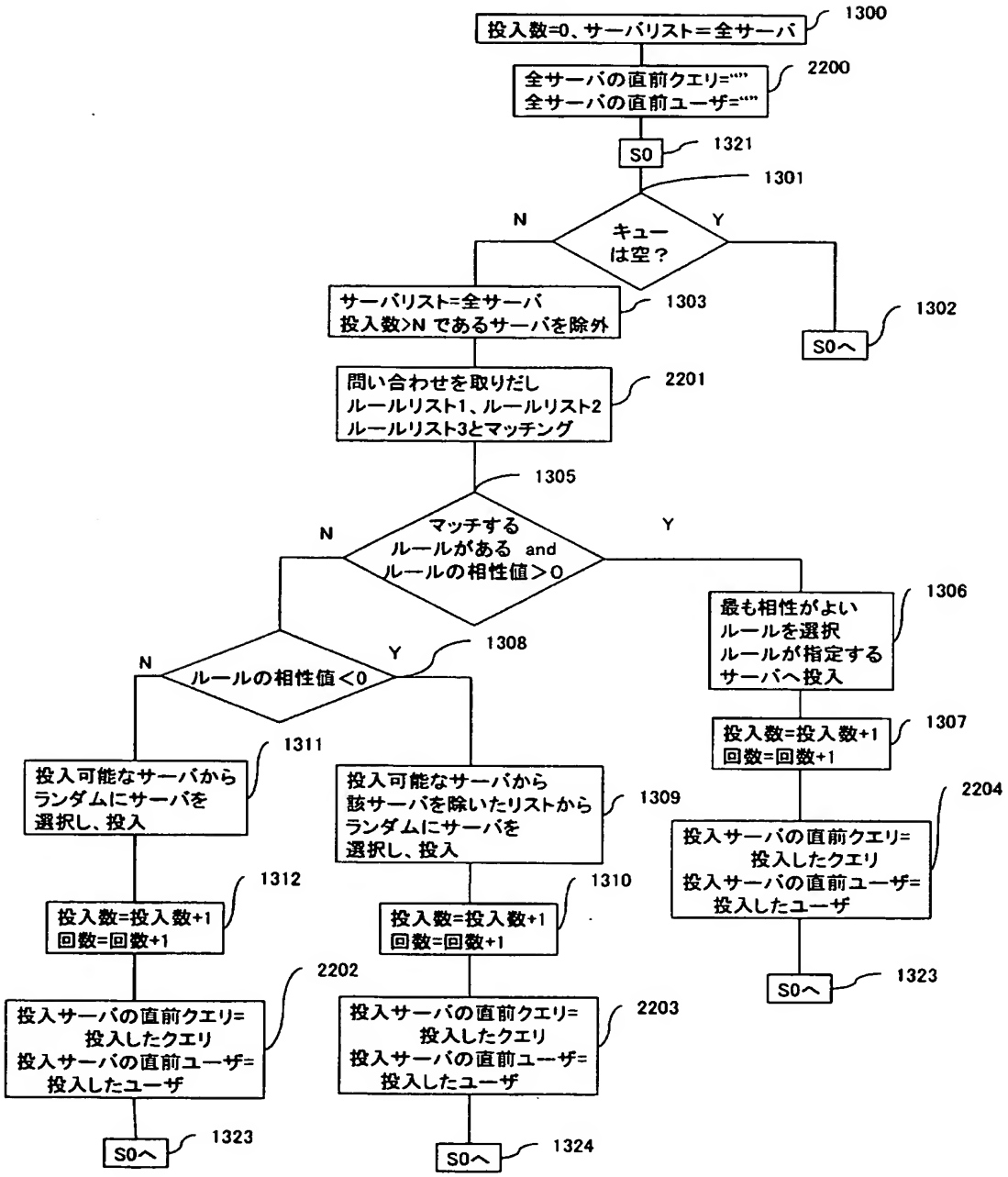


（C）ルールリスト3(ユーザ間の相性のルール)

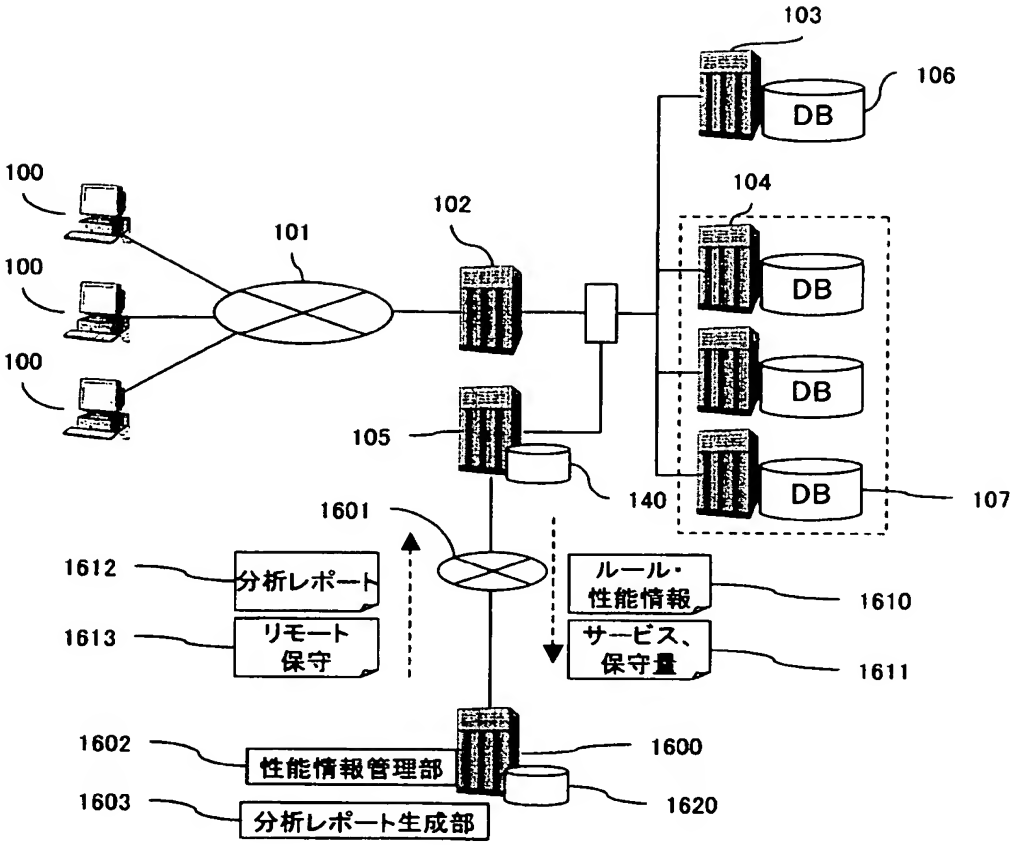


条件	相性値	回数
U1,U2	20	0
U1,U4	-10	0
...	...	...

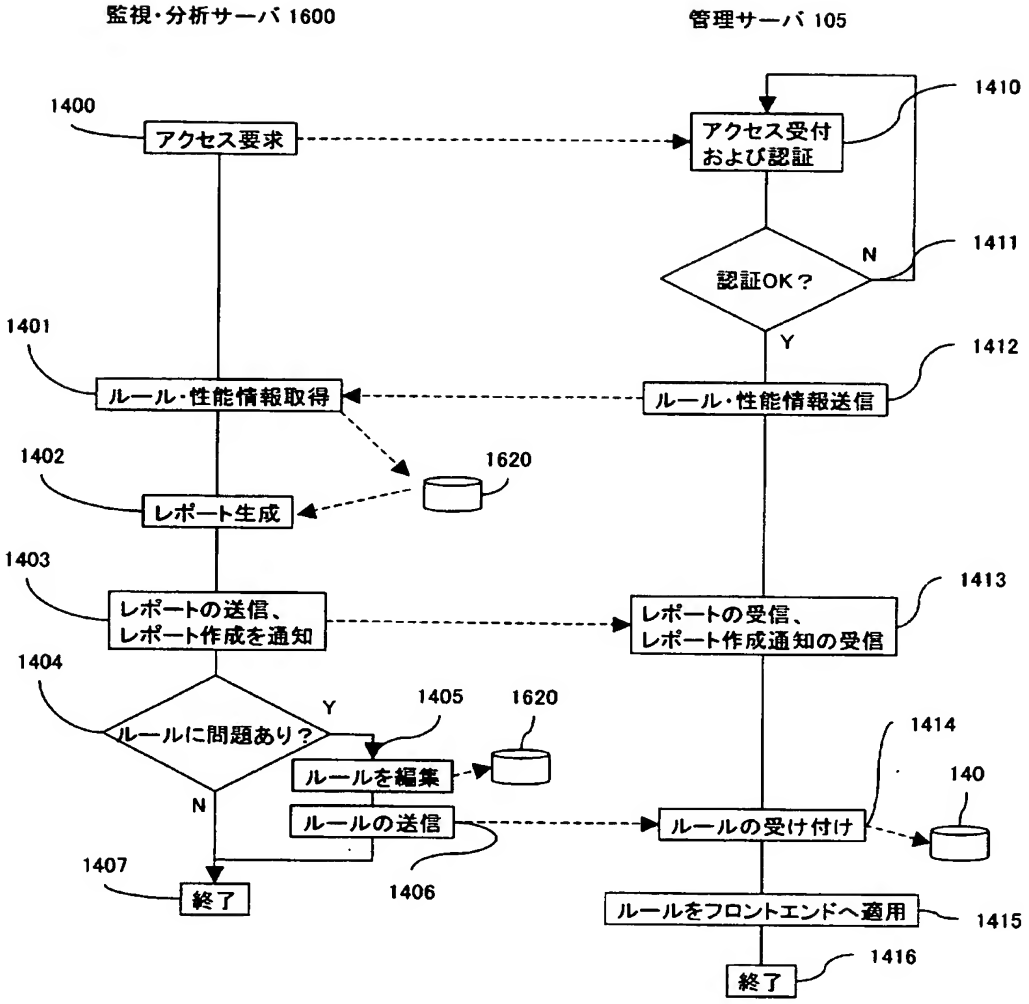
【図１６】


投入数=0、サーバリスト＝全サーバ
1300
全サーバの直前クエリ=""
全サーバの直前ユーザ=""
2200
S0
1321
1301
キュー
は空？
N
Y
サーバリスト=全サーバ
投入数>N であるサーバを除外
1303
S0へ
1302
問い合わせを取りだし
ルールリスト1、ルールリスト2
ルールリスト3とマッチング
2201
1305
マッチする
ルールがある and
ルールの相性値＞0
N
Y
最も相性がよい
ルールを選択
ルールが指定する
サーバへ投入
1306
ルールの相性値＜0
N
Y
1308
投入数=投入数+1
回数=回数+1
1307
投入可能なサーバから
ランダムにサーバを
選択し、投入
1311
投入可能なサーバから
該サーバを除いたリストから
ランダムにサーバを
選択し、投入
1309
2204
投入サーバの直前クエリ=
投入したクエリ
投入サーバの直前ユーザ=
投入したユーザ
投入数=投入数+1
回数=回数+1
1312
投入数=投入数+1
回数=回数+1
1310
S0へ
1323
2202
投入サーバの直前クエリ=
投入したクエリ
投入サーバの直前ユーザ=
投入したユーザ
2203
投入サーバの直前クエリ=
投入したクエリ
投入サーバの直前ユーザ=
投入したユーザ
S0へ
1323
S0へ
1324

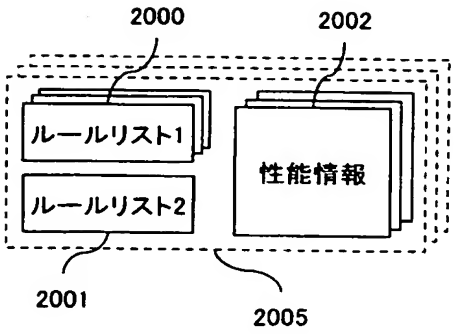
【図１７】


103
106
DB
100
100
100
101
102
104
DB
DB
DB
105
140
107
1601
1612
分析レポート
1613
リモート
保守
ルール・
性能情報
1610
サービス、
保守量
1611
1602
性能情報管理部
1600
1603
分析レポート生成部
1620

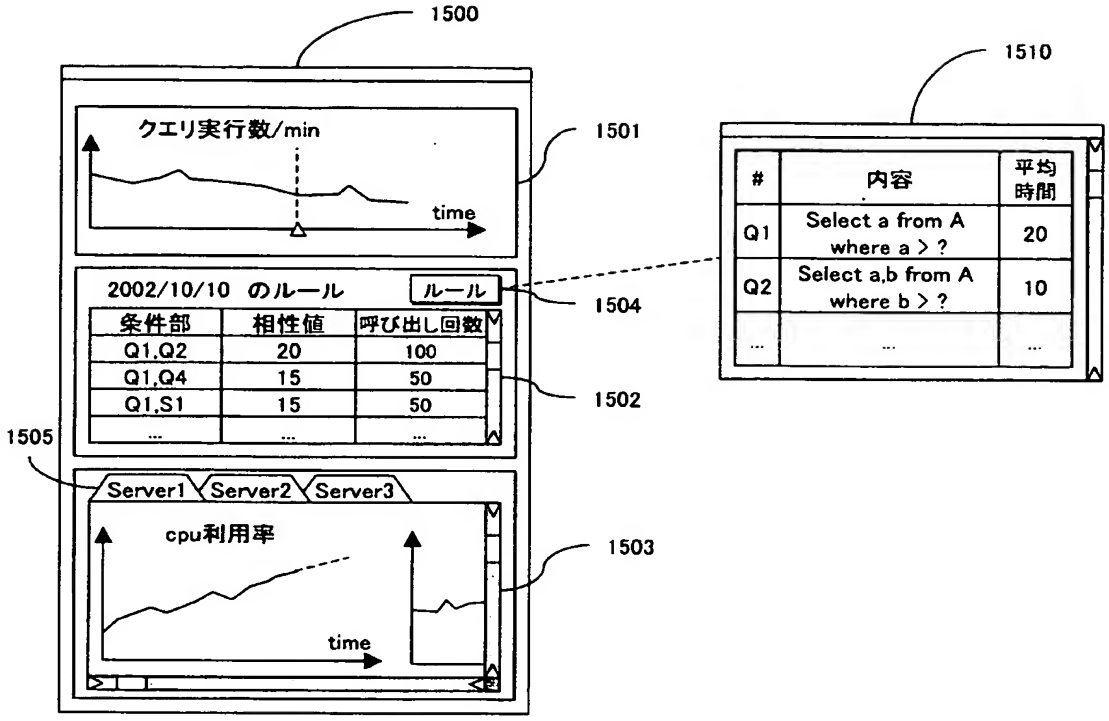
【図１８】


監視・分析サーバ 1600
管理サーバ 105
1400
アクセス要求
アクセス受付
および認証
1410
認証OK？
N
1411
Y
1401
ルール・性能情報取得
ルール・性能情報送信
1412
1402
1620
レポート生成
1403
レポートの送信、
レポート作成を通知
レポートの受信、
レポート作成通知の受信
1413
1404
ルールに問題あり？
Y
1405
1620
ルールを編集
1414
140
N
ルールの送信
ルールの受け付け
1407
終了
1406
ルールをフロントエンドへ適用
1415
終了
1416

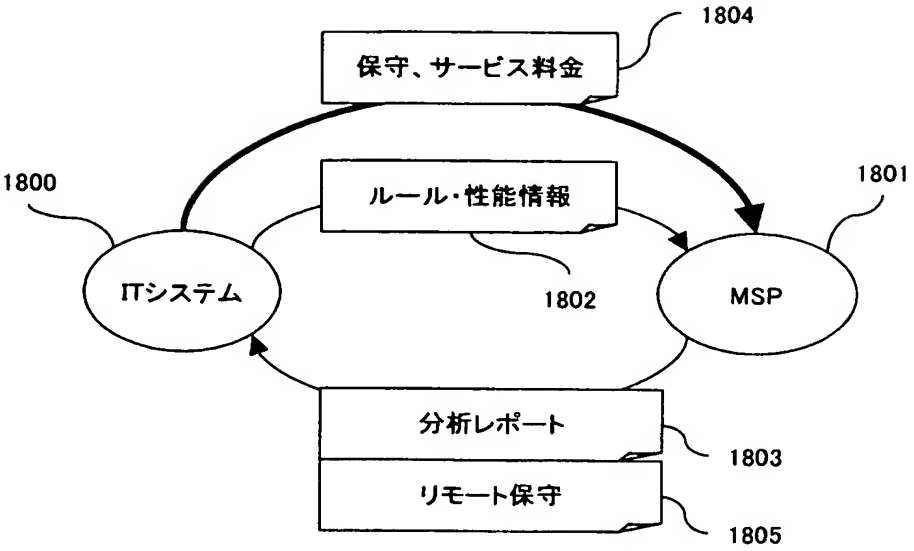
【図１９】



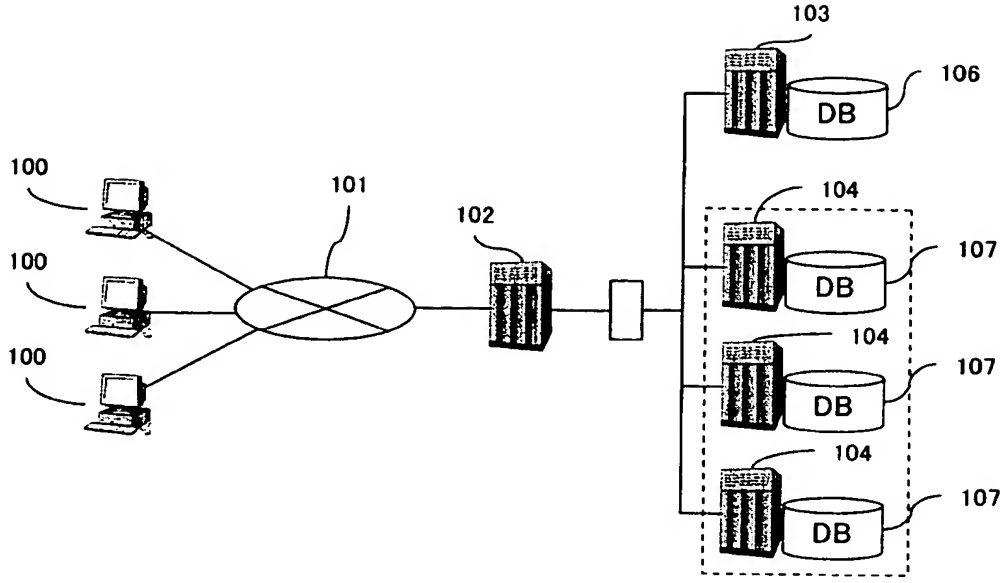
【図２０】


1500
1501
クエリ実行数/min
time
2002/10/10 のルール
ルール
1504
条件部
相性値
呼び出し回数
Q1,Q2
20
100
Q1,Q4
15
50
Q1,S1
15
50
1502
1505
Server1
Server2
Server3
cpu利用率
time
1503
1510
#
内容
平均時間
Q1
Select a from A where a > ?
20
Q2
Select a,b from A where b > ?
10

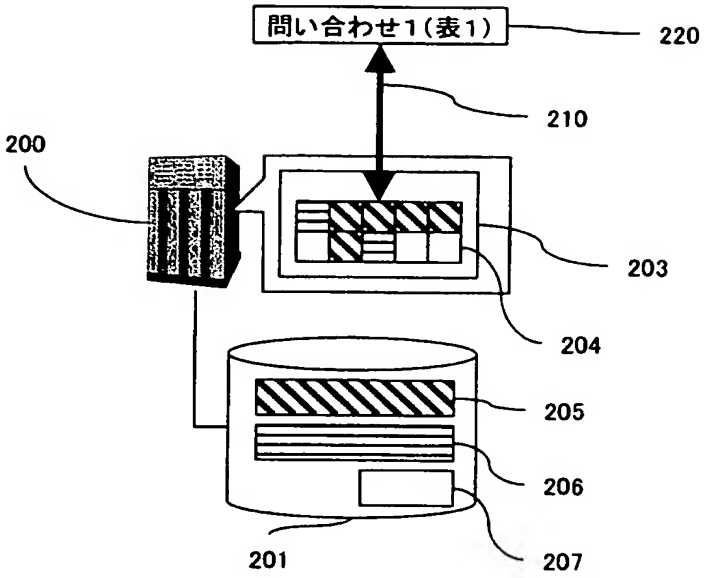
【図２１】


1804
保守、サービス料金
1800
ルール・性能情報
1801
ITシステム
1802
MSP
分析レポート
1803
リモート保守
1805

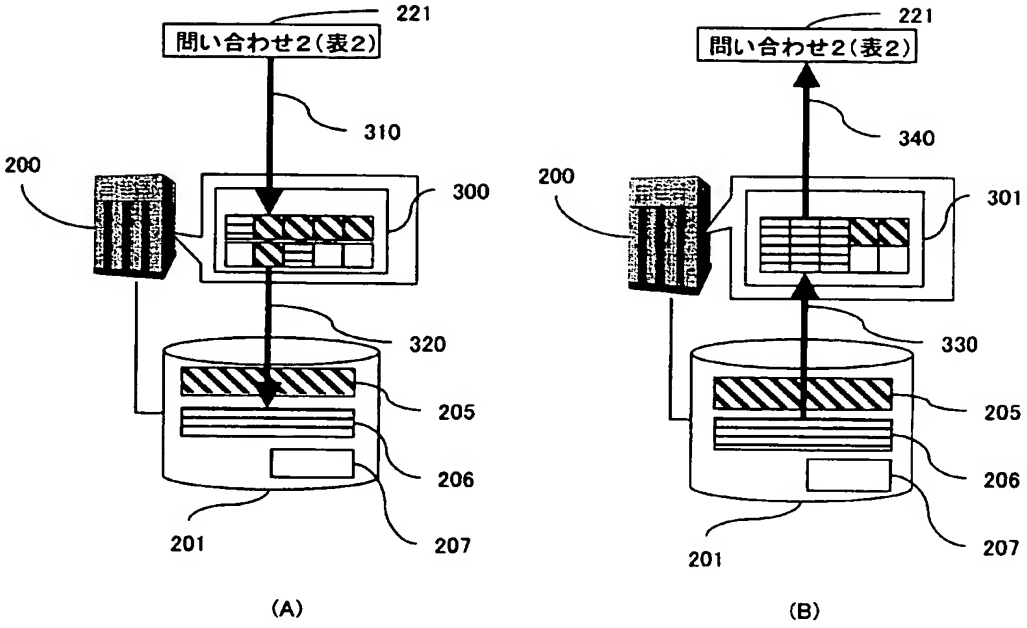
【図２２】



【図２３】



【図２４】


221
問い合わせ2(表2)
310
200
300
320
205
206
201
207
(A)
221
問い合わせ2(表2)
340
200
301
330
205
206
201
207
(B)

【書類名】　要約書

【要約】

【課題】　並列・分散型データベースシステムにおいて、リソース競合を回避し、システム全体の性能を向上させる。

【解決手段】　同一内容の検索が可能なデータベースを有し、問い合わせ要求に従って該データベースを検索する複数のデータベースサーバに対して、定められたルールを用いて問い合わせを投入するフロントエンドサーバと、前記フロントエンドサーバが使用するルールを管理する管理サーバと、を備えるデータベースシステムで用いられる問い合わせ投入方法において、前記管理サーバは、前記データベースサーバの実行ログを取得し、前記取得した実行ログを用いて計算された問い合わせに関する相性に基づいて前記ルールを生成し、前記フロントエンドサーバは、前記管理サーバで生成されたルールを用いて、前記問い合わせを投入することを特徴とする。

【選択図】　図１

特願２００３－０７１９０８

出　願　人　履　歴　情　報

識別番号　　［０００００５１０８］

１．変更年月日　　１９９０年　８月３１日
［変更理由］　　新規登録
住　所　　東京都千代田区神田駿河台４丁目６番地
氏　名　　株式会社日立製作所